Hierarchy Implosion for Faster User Equilibria on Road Networks[⋆]

Dennis Luxen and Peter Sanders

Karlsruhe Institute of Technology
{luxen, sanders}@kit.edu

Abstract. The road traffic of an entire day for a certain region can be understood as a flow with sources and sinks on the road network. Traffic has the tendency to evade regularly clogged roads and other bottlenecks, especially with modern on-board navigation devices that are able to interpret traffic information. In an ideal world, traffic would shape itself in a way such that all used routes between any two points on the road network have equal latency. Although these traffic patterns do not or very seldom occur in real life, they are a handy tool to predict the general traffic situation. For small networks, these patterns can be easily computed, but road networks that model entire countries are still a hurdle, because Dijkstras algorithm does not scale. Thus the known techniques have only been applied to either small networks or small extracts of a much larger network. We solve this problem for country sized road networks by combining a well-known formulation of traffic assignment as an optimization problem with current research on fast route planning by exploiting the special properties of a routing algorithm called Contraction Hierarchies. Our results show the feasibility of the approach for road networks that cover entire countries with running times that are suitable for a server-based implementation. Also, it shows that the actual size of the road network is not a limiting factor anymore.

1 Introduction

Traffic is often seen as a mere stream of cars. Consider the following picture. During a typical day of work traffic flows from the suburbs into inner cities in the morning and back from it in the evening. Or on national holidays a stream of cars and busses flows perhaps to resort towns or recreation areas close to the metropolitan areas. Naturally, some roads are more crowded than others since the traffic is not equally distributed over the road network. As a matter of fact, traffic has the natural tendency to shift itself to alternatives if it is more convenient for a driver to take the other route. Drivers seek to minimize travel time (or any other metric) and can be understood to act as selfish agents. They switch to better routes if they become aware of it. Assuming all drivers have full knowledge one is interested in how the traffic would distribute itself over the road network. This problem is known as the traffic assignment problem and is a major application in the field of transportation planning. Visually speaking, it is the process that finds edge latencies in a road network that are the result of many individuals competing for transportation.

[⋆] Partially supported by DFG grant SA 933/5-1

We assume travelers to take a least cost or (under some metric) shortest path between their origins and destinations. The problem at hand has been the subject of research since the early 1950s. Wardrops [29] first principle states the properties for a so-called *user equilibrium state*, which resembles the natural tendency of traffic to take a way of least resistance.

Definition 1 (Wardrops User Equilibrium) *A set of flows along the edges of a road network is said to be in a user equilibrium state (UE) when the two conditions of the following definition are met.*

1. *If two or more paths between the origin s and the destination t are actually traveled, then the cost of each path between s and t actually used must be the same.*
2. *There does not exist any path between s and t that is of less cost and unused.*

Finding a traffic pattern for which the above conditions hold is called *traffic assignment problem*. Solutions to this problem have a wide range of applications, for example in transportation management or in traveller information systems. Also, the real-time computation of equilibria states can be used as traffic forecasts and for traffic steering. Basic traffic jam avoidance is a feature of nowadays navigation devices. Unfortunately, this feature is not as developed as it is advertised.

Consider the following example. A traffic jam is reported for a certain highway and drivers on that highway are advised to leave their route by switching to an alternate road nearby. Since many drivers leave the highway, the road nearby is also clogged. This is not just an academic example, but happens every day. Germanys biggest automobile club ADAC reports in a large scale study [27] that most towns close to a highway suffer from increased pass-through traffic because of jam evaders. Routing on a road network that is at equilibrium is said to be a good estimate of routes that make not only economical sense but also are perceived as good alternatives to a clogged route. Todays jam evading features of navigation devices is limited. ADAC also reports a field study [6] that shows the inferiority of current approaches for traffic jam evasion. Not only the current traffic situation has to be considered to give better guidance around traffic jams, but also how the traffic will evolve.

Travelers on a road network are said to be non-cooperating. The state of the equilibrium is the aggregate result of individual decisions and therefore the name *user equilibrium*. It is generally assumed that under equilibrium conditions all used routes for the same origin destination pair have same costs, i.e. equal travel time. Also, unused routes between any origin destination pair have higher costs than used ones. Travellers are free to switch routes if there exists a better one than the current. The traffic distributes itself in a way that no traveller can lower its path cost unilaterally by switching to a cheaper path. This is the case at equilibrium, because by definition there is simply no such path. Note that this equilibrium state is closely related to the conditions of a Nash Equilibrium.

A general behavioral assumption in the field of transportation science is that each traveler or vehicle in a road network will take a path that has least cost (or is at least perceived as such). It is further assumed that travel time is the most significant utility for route choice. We recognize the over-simplification of this model, but direct the reader to the literature on empirical research of route choice, i.e [22]. We stick to travel time as edge cost or distance measurement throughout this paper.

The remainder of this paper is organized as follows. First, we look at the relevant literature in Section 2. We introduce the basic algorithms and data structures that we use and explain how they solve the problem at hand in Section 3. Second, we present an experimental evaluation in Section 5 that shows the performance of our approach. The method is applied to a graph that models the entire road network of Belgium and Germany. Section 6 summarizes the results and presents future directions of research.

2 Related Work

The traffic assignment problem (TAP) has been studied for more than 50 years. The first mathematical formulation is generally attributed to Beckman et al. [4] and was first given in 1956. It formulates the traffic assignment problem as an equivalent optimization problem. See Appendix **??** for a formal definition of the problem.

The method of choice to solve this problem is the Frank-Wolfe algorithm [17], which is also known as the *convex combinations algorithm*. It was originally invented to solve quadratic programming problems. Over the years it has been applied to the traffic assignment problem, mainly because of its rather simple structure. Occurences in the literature go back to the late 1960s [7,18]. The major advantage of the Frank-Wolfe algorithm (besides it's simplicity) is its low memory consumption. For example, it does not save any information on computed routes. It only counts the volume of traffic on each individual street segment. This was considered a major advantage in the early days of computation, because of limited memory capabilities. The algorithm alternates between an assignment phase of the traffic demand and a minimization step to numerically approximate edge flows.

The textbook of Sheffi [24] gives an overview of the first three decades of research between 1950 and 1980. Most of the solution techniques described are still in use by practitioners today. Usually they are applied to road networks of small and medium size up to several hundred or a few thousand edges and often only on sparse subsets of highway networks which are much smaller than the full road network.

There are several publications that focus on speeding up convergence of solving the traffic assignment problem by modifying the way traffic flow is distributed during the computation. Gentile [13] proposes an algorithm that seeks a deterministic equilibrium for the local route choice of users directed toward a same destination at every node. Bar-Gera [1] presents an algorithm to compute the UE by paired alternative segments. If flow between two nodes splits into separate subpaths than flow is shifted proportionally.

A completely different model to solve the traffic assignment problem is to apply game theory. Rosenthal [23] was the first to consider the problem by a game theoretic approach. A so-called congestion game is defined by a set of players that compete for one or more shared resources. It is said to be symmetric if all players chose among the same set of strategies. Fabrikant et al. [11] show that any symmetric congestion game can be solved in polynomial time. Relating to our case the players are travellers that compete for roads and seek to minimize travel expenses. Edge latencies, i.e. the time time it takes to traverse a road segment, are defined to be nonnegative, continuous and nondecreasing functions of the amount of travellers on that edge. A potential is defined by summing over the edge latencies of a solution under which each traveller has chosen a route. This potential can be easily optimized to a (local) minimum by allowing players to switch their strategy, which is a shortest route in this case. These switches are called selfish steps. Consider a move of one of the players to a better route. Any local optimum corresponds to the conditions stated in Definition 1. It is easy to see that the potential is lowered and that it can be brought to a minimum by subsequent switches until no switch to an improved route for any player is possible.

Kirschner et al. [16] apply book keeping heuristics to avoid many path computations and subsequently speed up the rate of convergence on networks with less than a few thousand nodes and edges. For an excellent survey over the literature for congestion games and algorithmic game theory in general see the textbook of Nisan et al. [19]. Note that the game theoretic approach prohibits any precomputation that exploits the underlying network topology, i.e preprocessing that is done for a fixed metric only. A single selfish step might change the topology enough to invalidate the preprocessed data structures and preprocessing the network for a single query is out of the question.

The application of the Frank-Wolfe algorithm and also the game theoretic solution need a method of path finding. Plain solutions spend virtually all of the computational effort in path finding. Unfortunately, Dijkstras algorithm does not scale well on large road networks with millions of nodes and egdes. For large scale applications [20] this is simply inacceptable. Therefore speedup techniques for point-to-point queries with Dijkstras classic algorithm have been the focus of numerous publications before. For surveys on the literature and combinations of several methods see [10,3].

Contraction Hierarchies [12] is a very successful speedup technique that has the advantage of combining a simple algorithmic concept and very good speedups. The technique is based on the concept of contracting nodes. The nodes of the input graph $G = (V,E)$ are ordered by some measurement of importance. Unimportant nodes are bypassed and replaced by so-called shortcut edges to preserve shortest path distances The resulting data structure can be queried by a bidirectional Dijkstra to find shortest paths. We refer the interested reader to the publication of Geisberger et al. [12] for an in-depth explanation of the node ordering and proofs of correctness. There have also been reports on combinations of several distinct speedup techniques [3].

To the best of our knowledge there is no publication that reports on directly exploiting the special properties of such a speedup technique to augment traffic assignment computations. Also, we are not aware of equilibria computations for large networks with significantly more than a few hundred or thousand street segments [14,1,13].

3 Problem Formulation

We model a road network as a graph $G = (V,E)$. V is a set of nodes and $E \in V \times V$ is a set of edges or less formally the set of street segments. Each edge carries a certain amount of traffic that we call flow. Each edge e is labelled with an edge weight $w_e = c(f_e)$ that is the result of the flow f_e on e and edge cost function c. Given n nodes in a network, let nodes $1,\dots,p \leq n$ be a subset of nodes which are either origin or destination of a so-called demand set.

Generally, we view the nodes of the graph as the places where traffic passes-by, enters or leaves the system. We define the set of demands D as a set of triples (i, j, k), where $i, j \in V$ and $k \in \mathbb{N}$. The nodes i and j indicate origin and destination nodes and k the number of units that demand to flow between these nodes. Note that demands are integral. Flow on a certain road segment is said to be the ratio of the current and maximum number of vehicles on that segment. The maximum flow of a road segment is the ratio of length of the segment times the number of lanes and the average length of a car. In absence of data we defined motorways and motorway links to have two lanes while the other categories feature only one lane.

The traditional name OD-matrix for the demand set comes from the case where the number of modeled nodes equals the number of all origins and destinations with flow going from every node to every other node. One reason for this name might be the result of the shortest paths algorithms itself. As we argued before, Dijkstras algorithm does not scale well on large instances. Finding the shortest path for each and every origin-destination pair is a heavy computational burden on large graphs. So it is more efficient to model each node as an origin of traffic and to compute the shortest path search tree for each node exactly once and to store the resulting distances in the column of a matrix. Multiple destinations will be found with just one search. As we show in this and later Sections, this model is not imperative anymore.

Optimization Problem. In [24] it has been shown that the traffic assignment problem can be solved as a minimization problem. The objective function of the underlying optimization problem is based on total edge flows and the resulting edge weights. Consider ω to be the flow on an edge. The function is defined by the sum over the change of all edge weights

$$\min(\mathbf{z}) = \sum_{e \in E} \int_0^{f_e} C_e(\omega)\, d\omega$$

with the constraint, that the sum over all observed flows between any two nodes equals the total demand between those nodes. This minimization problem can be solved by applying the Frank-Wolfe algorithm [17,24]. In each step of the algorithm the approximation of the solution is replaced by a new approximation that is obtained by gradient descent towards the optimum.

Initialization and Iterative Improvement. The initialization is an all-or-nothing assignment of the demand set where each demand is assigned to the edges of the shortest paths using free flow speed on the edges. In other words, travellers choose the routes that would be best if they were the only travellers on the road network. These free flow usages are counted and edge weights reevaluated w.r.t. the flow on the edges and these edge weights are taken as the initial solution X^0. In each iteration a subsequent assignment Y^i is computed and combined with the previous solution to get a better approximation.

More formally, the n-th iteration starts with an update of all edge weights $C_e = (c_1^n, \dots, c_{|E|}^n)$ corresponding to the edge flow vector $F^n = (f_1^n, \dots f_{|E|}^n)$ which is the result of the previous iteration. Next, an all-or-nothing assignment that distributes the so-called auxiliary flow $Y^n = (y_1^n, \dots y_{|E|}^n)$ on the network is performed. The new approximation

$$X^{n+1} = X^n + \alpha^n \cdot (Y^n - X^n)$$

is obtained by computing a scaling factor α^n that is feasible in the current iteration only. Note that computing Y^n is straight-forward and X^n is known from previous iteration. We solve

$$\alpha^n = \min_{0 \le \alpha \le 1} \sum_e \int_0^{f_e^n + \alpha(y_e^n - f_e^n)} C_e(\omega) d\omega$$

at each iteration. Since we know the derivative of the function, we can solve that step with a search strategy to find the minimum. This is also known as line search.

The search for α^n is solved approximately with a certain error threshold by applying the bisection method of Bolzano, which finds the zero of a continuous function by a recursive descent similar to binary search. For any given interval $[a,b]$ and $c = (b+a)/2$ we examine if our solution is either in $[a,c]$ or $[c,b]$ and decent recursively until we have reached a certain accuracy. Note that the number of repeated bisections $\mathscr{N}$ needed to approximate $\alpha \in [0,1]$ with an error less or equal than a $\delta > 0$ is given by $\mathscr{N} := -\log_2 \delta$.

The series of solutions $X^i, i > 0$, is known to converge to the solution of the traffic assignment problem. Again, we refer the reader to the textbook of Sheffi [24] for in-depth explanations and for the correctness of the method.

Edge Cost Functions. If the travel time between any two nodes was a constant independent of the flow in between then we could solve the problem easily. It would suffice to compute the shortest path for each element of the demand set. Of course, this view neglects reality and the effect that flow, or in other words dense traffic, has to the average speed on a road segment. The denser the traffic gets the more careful drivers have to be not to cause an accident by running into a decelerating car in front. Likewise the denser the traffic the more cars are affected by ones own driving maneuvres [26].

To model the situation more realistically the edge cost function has to be increasing, contiguous and non-linear. Several good edge cost functions have been proposed. A simplified function is the Bureau of Public Roads [8] function (BPR). This function was derived from empiric observation and takes length, speed limit and capacity as parameters. Although it is easy to compute its curves are not asymptotic to any maximum capacity value, which is in stark contrast to reality. To overcome this shortage Davidson [9] proposed a function family that is based on queuing theory. It is defined as

$$t_e = t_e^0 \cdot \left[1 + J \cdot \frac{x_e}{c_e - x_e} \right]$$

where t_e^0 denotes the travel time at 0 usage, c_e the capacity of the street segment and x_e the current usage. J is a tuning parameter to control the shape of the curve.

Other classes of road functions have been proposed. For example, the class of *conical volume-delay functions* [25]. For an earlier survey on edge cost functions see [5]. But on the other hand the Davidson function models basic relationships between usage und resulting travel times and it is easy to compute. We set J to 0.25 throughout this paper. See Appendix **??** for a plot of the Davidson function function family for different values of J showing the relation between average travel times and the amount of flow relative to maximum flow.

Convergence Criterion. Convergence can be based on a number of criteria. Clearly, one would like to stop once the changing of edge weights comes to a halt between iterations. The easiest choice is to stop after a fixed number of iterations, but this entirely neglects solution quality. A natural choice would be to use the change of the objective function as convergence test. This might be misleading, since the lengths of individual paths might differ significantly while the sum of of the lengths is relatively stable. Therefore, the stopping criterion is based on how much the path length for each demand differs between two iterations.

$$\max_{d \in D} abs\left(\frac{\mu^n(d) - \mu^{n-1}(d)}{\mu^{n-1}(d)}\right)$$

where

$$\begin{aligned} D &= \text{set of demands} \\ \mu^n(d) &= \text{length (cost) of path in iteration } n \text{ for demand } d \end{aligned}$$

This stopping criterion indicates the quality of the approximation of the equilibrium much better from a behavioral point of view than a simple sum of all edge weights. Furthermore it ensures that the computation is only stopped once the weights of all edges have settled down.

4 Integration into Contraction Hierarchies

A naive implementation of the optimization algorithm of Section 3 is technically easy and straight-forward with any algorithm that computes shortest path, i.e. Dijkstras Algorithm. A more efficient approach will be explained in this Section.

As we mentioned before, it is not necessary to save any information on the paths that are computed. After all path computation is done in an iteration, it suffices to know how often each edge occurs in the set of shortest paths that are computed. So, it is a characteristic of the optimization that there is no need at all for saving the path. Although, computing and unpacking each path is technically feasible, it is also possible to integrate the path computation of Contraction Hierarchies more efficiently.

Note that any shortest path that is computed by the bidirectional Contraction Hierarchies query on the search data structure consists of shortcuts. Although the length of any shortest path is optimal, it has to be unpacked to actually know which edges of the original graph are used. Usually, unpacking is done by a recursive method. The edges of the packed path are pushed onto a stack and while the stack is non-empty an edge is popped. If it is a shortcut then the two edges building that shortcut are pushed onto the stack. Otherwise the popped edge is inserted into the resulting unpacked path. The recursive unpacking runs fast in time linear to the length of the unpacked path. When compared to a plain Dijkstra algorithm unpacking is still several orders of magnitude faster. But with the help of the next observation, we can do even better.

Consider the case that the method of Section 3 has been implemented naively with a variant of Dijkstras algorithm. Each (original) edge of the road network needs to keep track of the number of times it occurs in a path. From our experience from the experiments of Section 5 and also from a brief look into reality we can tell that many street segments of a road network, i.e. highways and other roads of high importance, are traversed a vast number of times, if the demand set is sufficiently large. So, many single edges will be touched several times during the computation and this is not necessary as we see in the remainder of this Section.

We modify the path computation in a way that each shortcut is unpacked exactly once during each assignment phase. At first, we do not unpack the paths at all, but count the flows on edges without unpacking shortcuts. To do so, each edge is equipped with a counter to record the number of times it is part of a shortest path. This number is counted during the path computation. It can be done easily, since each path consists of a few shortcuts only. Recall that we do not need to keep track of the routes actually chosen by travellers, but only the amount of flow on each individual edge. After all paths have been computed the hierarchy is deconstructed by unpacking all shortcuts and assigning the load of the shortcut to edges that lie underneath. See Figure 1 for an illustration of the process of hierarchy implosion.

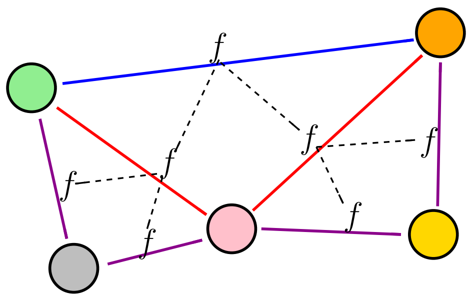


Fig. 1. Usage gets distributed through dashed edges to edges of shortcuts

The only prerequisite to the correctness of this approach is to implode the shortcutes in the opposite order in which they were created during creation of the hierarchy. It is obvious from the order of implosion that no shortcut needs to be unpacked more than once and no edge usage is lost. The order of shortcut creation is easy to record during preprocessing and takes up a neglectable amount of space only. The reverse order of insertion defines the order of implosion. Note that the order can also be determined by a topological search since the hierarchy is a directed acyclic graph.

If each path would be unpacked directly each time a path gets computed then the heavily traversed edges would be touched many, many times to increase usage counters. Instead, the path unpacking is independent of the number of demands and depends only size of the hierarchy. Note that the implosion of the hierarchy is not recursive.

Our resulting algorithm performs very well as we will see in Section 5. The additional space overhead for shortcut order and edge usage counters is more than bearable on a current desktop computer.

4.1 Demand Generation

We pregenerate randomized lists of origin destination pairs, also called *the set of demands* or *demands* for short, for the test cases of Section 5. It is out of the scope of this paper to generate demand sets that reproduce observed conditions from reality. A simple trip generation model is presented that generates traffic demand that is realistic enough to show the validity of the technical approach.

The set of demands is fixed and therefore does not change during the assignment. To the best of our knowledge we are not aware of any high resolution trip generation algorithms coming from transportation science that covers entire countries. Also, we do not aim to provide a completely realistic simulation, but rather a tool to demonstrate the technical feasibility of the approach. As a consequence, we have to resort to trip generation that does not claim the character of any close approximation of reality.

During a personal conversation with an ADAC representative it was suggested to us that the distances actually traveled are geometrically distributed with an expected distance of 40 kilometres. We conjecture that the population density correlates strongly with the density of a road network and choose the starting points uniformly and at random from the set of all nodes. Since we know the distribution, we draw a geometrically distributed distance that each lies between each start and target node. A ball is grown around each starting node s using a unidirectional Dijkstra Search and when an edge is relaxed we check the distance its end node has from the source. If the distance of the end node is equal or more than the travel distance that was drawn before, we accept the node as the target t of s and insert the pair (s,t) into the demand set.

5 Experimental Evaluation

We implemented our algorithm and data structures in C++ using GCC v4.3.2 as compiler with full optimizations turned on. All tests were done on a single core of a Intel Xeon X5550 CPU running at 2.67GHz. The machine is equipped with 48 GB of RAM running Linux kernel version 2.6.27. The evaluation was done on road networks of Belgium and Germany. The Belgian network consists of 463 514 nodes and 1.093 454 edges whereas the german network consists of 4 378 446 nodes and 9 574 254 million edges. Both have been made available by PTV AG[1] for scientific use. For each road segment length and the respective out of 13 road categories are available. Free flow speeds have been derived from category and length of an edge. Capacity is implied by the category of the edge, which is an oversimplification, but unavoidable because of lack of data. Travel times were computed according to the Project OSRM car speed profile [21].

We pregenerated lists of $10^5, 10^6$ and 10^7 demands for the network of Belgium and Germany and in addition also a list of 10^8 demands [2] for Germany to reflect the larger size of the road network. We computed the user equilibria for all demand sets on the respective graphs. The line search approximation parameter was set to 10^{-10} and the

[1] http://www.ptv.de

[2] The experiments 10^8 will appear in the final version only, because of hardware problems just before the deadline.

dampening factor J of the davidson edge cost function was set to 0.25. See Figure 5 for the numerical results of the experiments.

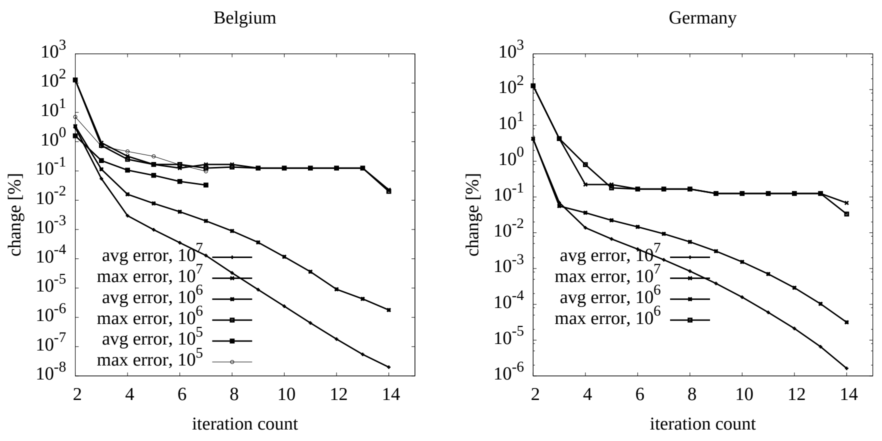


Fig. 2. Experimental results for the Belgian (left) and German (right) road network.

The stopping value is quickly approached in each of the experiments. We observe that the stopping value of a maximum error of less than 0.001% is approached in a similar way for each of the demand sets while the average error drops significantly with larger demand set sizes. The larger the graph and the demand set the more evident this phenomenon gets.

The traffic assignment changes the topology of the underlying graph. This ist a direct consequence of Wardrops User Equilibrium from Definition 1. Recall that under equilibrium state all used routes for a certain origin-destination pair have equal travel times. The edge weights are adapted iteratively to (approximately) reach this state. This flattens the natural hierarchy of the road network that is exploited during the contraction phase. Thus the preprocessing takes longer, because it is harder to decide if a certain shortcut is needed or not. Less shortcut edges can be omitted from the search data structure, because for many shortcuts there is now a shortest path that actually lies on it. Likewise, the query times rise. The effect is most obvious in the first two iterations when the most changes occur in the edge weight. See Appendix A and B for plots for all the experiments.

From the plots of Appendix A and B it becomes clear, that the larger the numbers of queries the less important preprocessing and the road network size gets.

Note we also implemented a simpler iterated all-or-nothing assignment. The method starts with a feasible flow on the network. Then edge costs are recalculated for the flow, which is observed on each edge. The flow is reassigned to the changed network and the process is reiterated until a specified number of iterations is completed. We did not observe any convergence with this technique even for large numbers of iterations. In contrast, we observed oscillation of route choice and quickly deemed the approach infeasible. Likewise, an incremental loading where a subset of the demands is assigned proved infeasible as well. Again, convergence did not occur.

6 Conclusions

Our algorithm exploits the special properties of the search data structure of Contraction Hierarchies which enables us to solve the problem with better efficiency than pure path computation with unpacking of each computed path. We showed the feasibility of large scale traffic assignment on graphs that cover entire countries. Although our algorithm works sufficiently well for smaller countries like Belgium, there is still some room for improvement. The running times for Germany are not yet fast enough to allow near-realtime traffic assignment.

A parallel implementation of our algorithm is straight-forward. Vetter has already implemented a parallel Contraction Hierarchies variant [28] that could be adapted for hierarchy implosion. We are working to extend our research to time-dependent road networks, multiple cost functions and also distributed computation. Contraction Hierarchies have already been adapted to time-dependent road networks [2] and there has been a distributed implementation recently [15] that can be used to speed up the pre-processing phase even further. Finding the right modelling of the time-dependent traffic assignment problem is an interesting question on its own.

References

1. Bar-Gera, H.: Traffic assignment by paired alternative segments. Transportation Research Part B (2010)
2. Batz, V., Delling, D., Sanders, P., Vetter, C.: Time-Dependent Contraction Hierarchies. In: Proceedings of the 11th Workshop on Algorithm Engineering and Experiments (ALENEX'09). pp. 97–105. SIAM (2009)
3. Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., Wagner, D.: Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm (2009), ACM JEA special issue for WEA 2008
4. Beckmann, M., C.B.McGuire, C.B.Winsten: Studies in the economics of transportation. Yale University Press, New Haven (1959)
5. Branston, D.: Link capacity functions: A review. Transportation Research 10(4), 223–236 (August 1976)
6. Brieter, K., Eicher, C.C., Haart, V., Vigl, M.: Mit dem navi sicher in den stau. ADAC Motorwelt 3, 56–59 (2010)
7. Bruynooghe, M., Gilbert, A., Sakarovitch, M.: Une methode d'affectation du trafic. In: Leutzbach, W., Baron, P. (eds.) Proc. 4th Internat. Sympos. Theory Road Traffic. pp. 198–204. Bundesminister für Verkehr, Abt. Strassenbau, Bonn, Germany (1969)
8. Bureau of Public Roads: Traffic Assignment Manual. U.S. Dept. Of Commerce, Washingtion D.C. (1964)
9. Davidson, K.B.: A flow travel time relationship for use in transportation planning. Proceedings of the Australian Road Research Board 3, 183–194 (1966)
10. Delling, D., Sanders, P., Schultes, D., Wagner, D.: Algorithmics of Large and Complex Networks, chap. Engineering Route Planning Algorithms, pp. 117–139. Springer (2009)

11. Fabrikant, A., Papadimitriou, C., Talwar, K.: The complexity of pure nash equilibria. In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. pp. 604–612. STOC '04, ACM, New York, NY, USA (2004)
12. Geisberger, R., Sanders, P., Schultes, D., Delling, D.: Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: McGeoch, C.C. (ed.) WEA. Lecture Notes in Computer Science, vol. 5038, pp. 319–333. Springer (2008)
13. Gentile, G.: Linear user cost equilibrium: a new algorithm for traffic assignment. submitted for publication in Transportation Research B (2010)
14. Jayakrishnan, R., T.Tsai, W., Prashker, J.N., Rajadhyaksha, S.: A faster path-based algorithm for traffic assignment. Transportation Research Record (1994)
15. Kieritz, T., Luxen, D., Sanders, P., Vetter, C.: Distributed time-dependent contraction hierarchies. In: Proceedings of the 9th International Symposium on Experimental Algorithms (SEA 2010). Springer (2010)
16. Kirschner, M., Schengbier, P., Tscheuschner, T.: Speed-up techniques for the selfish step algorithm in network congestion games. In: Proceedings of the 8th International Symposium on Experimental Algorithms. pp. 173–184. SEA '09, Springer (2009)
17. Marguerite, F., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quaterly 3, 95–110 (1956)
18. Murchland, J.: Road network traffic distribution in equilibrium. Mathematical models in the social sciences (1979)
19. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)
20. Patriksson, M.: The Traffic Assignment Problem: Models and Methods. VSP, Utrecht, The Netherlands (1994)
21. Project OSRM: http://project-osrm.org, http://project-osrm.org
22. Ramming, S.M.: Network Knowledge and Route Choice. Ph.D. thesis, Massachusetts Institute of Technology (February 2002)
23. Rosenthal, R.W.: The network equilibrium problem in integers. Networks 3, 53–59 (1973)
24. Sheffi, Y.: Urban Transportation Networks: Equlibrium Analysis with Mathematical Programming. Prentice-Hall, Inc Englewood Cliffs, NJ 07632 (1982)
25. Spiess, H.: Technical Note–Conical Volume-Delay Functions. TRANSPORTATION SCIENCE 24(2), 153–158 (1990)
26. Sugiyama, Y., Fukui, M., Kikuchi, M., Hasebe, K., Nakayama, A., Nishinari, K., ichi Tadaki, S., Yukawa, S.: Traffic jams without bottlenecks – experimental evidence for the physical mechanism of the formation of a jam. New Journal of Physics 10(3), 033001 (2008)
27. Unknown Authors: TMC-Stauumfahrung:Verkehrsprobleme durch Stauverlagerungen? Tech. rep., ADAC e.V. (2010)
28. Vetter, C.: Fast and Exact Mobile Navigation with OpenStreetMap Data. Master's thesis, Karlsruhe Institute of Technology (2010)
29. Wardrop, J.G.: Some theoretical aspects of road traffic research. In: Proceedings of the Institute of Civil Engineers. vol. 1, pp. 325–378 (1952)

A Running times for experiments on Belgian road network

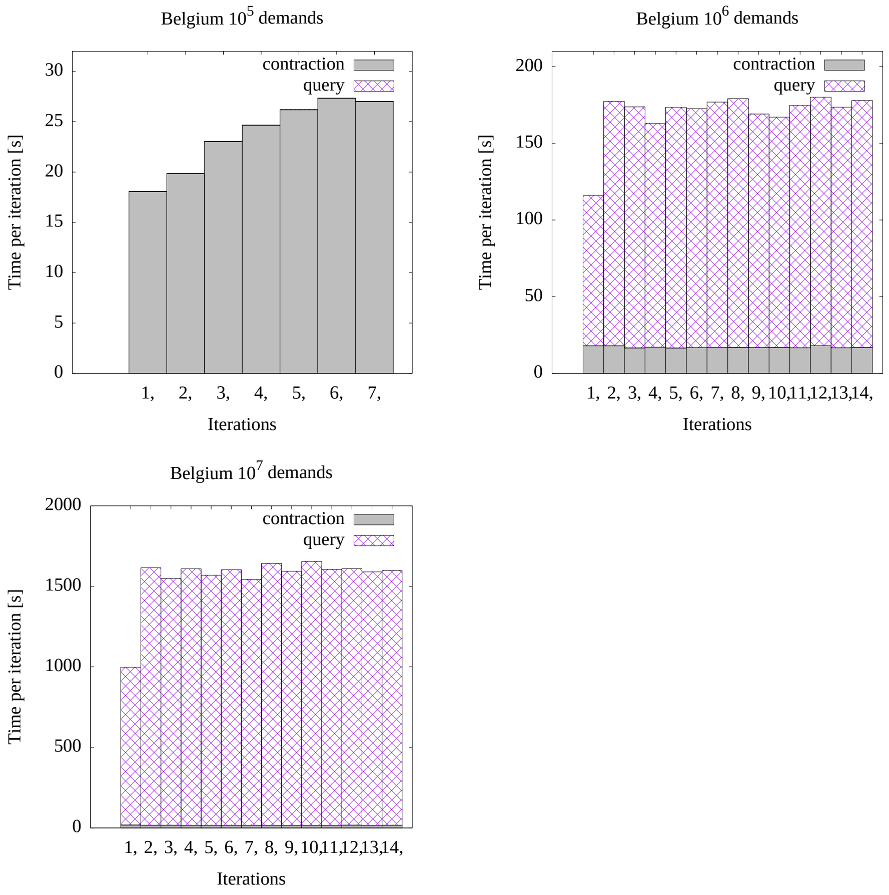
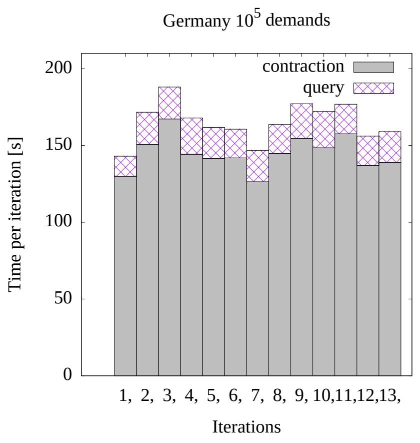
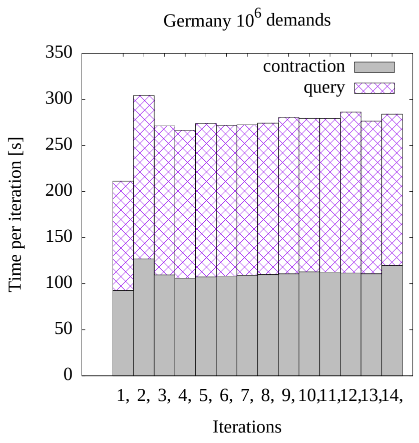


Fig. 3. Running times for Belgian network

B Running times for experiments on German road network





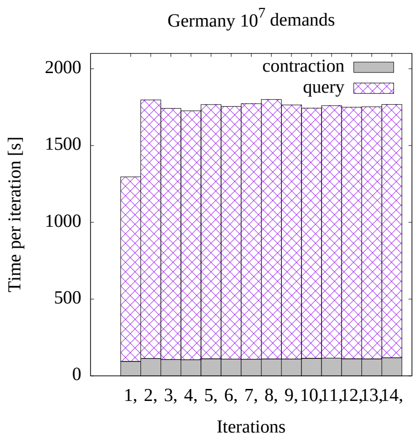


Fig. 4. Running times for German network

C Mathematical Programming notation of the Traffic Assignment Problem

This formulation of the problem is due to Beckman Beckman et al. [4].

$$\begin{aligned}
\min(\mathbf{z}) &= \sum_{e\in E}\int_0^{f_e} c(\omega)\,d\omega \\
\text{s.t. } f_e &= \sum_r\sum_s\sum_p h_p^{rs}\delta_{e,p}^{rs} \quad \forall e\in E \\
\sum_p h_p^{rs} &= q^{rs} \quad \forall r\in R, s\in S \\
h_p^{rs} &\geq 0 \\
\delta_{e,p}^{rs} &= \begin{cases} 1 & \text{if path } p \text{ contains edge } e \\ 0 & \mathit{else} \end{cases}
\end{aligned}$$

E,R,S	$\hat{=}$ edges in road network, set of origins and destinations
Z	$\hat{=}$ numerical value of the objective function
f_e	$\hat{=}$ total flow on edge e
$C_e(f_e)$	$\hat{=}$ cost of traversing edge e under flow
h_p^{rs}	$\hat{=}$ flow on path p from $r\in R$ to $s\in S$
q^{rs}	$\hat{=}$ number of trips going from $r\in R$ to $s\in S$

D Plot of the Davidson function family

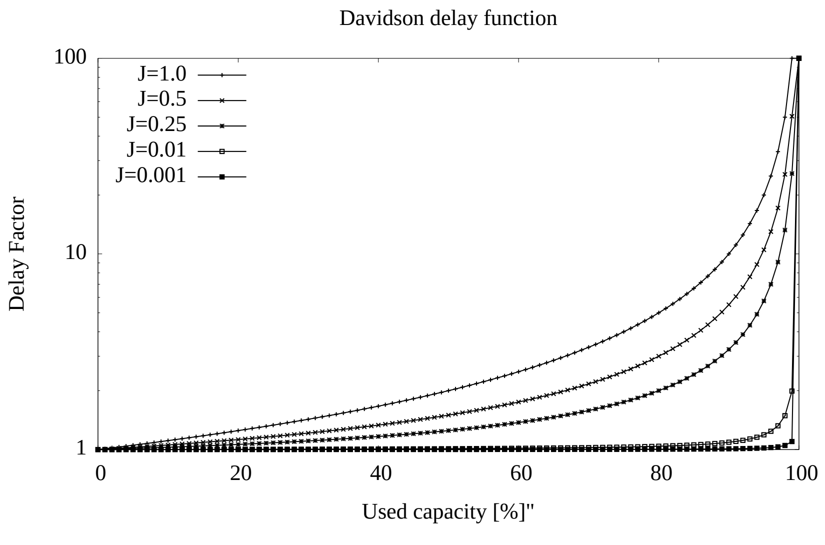


Fig. 5. Plot of the BPR function

E Plots of Germany and Belgium at Equilibrium state

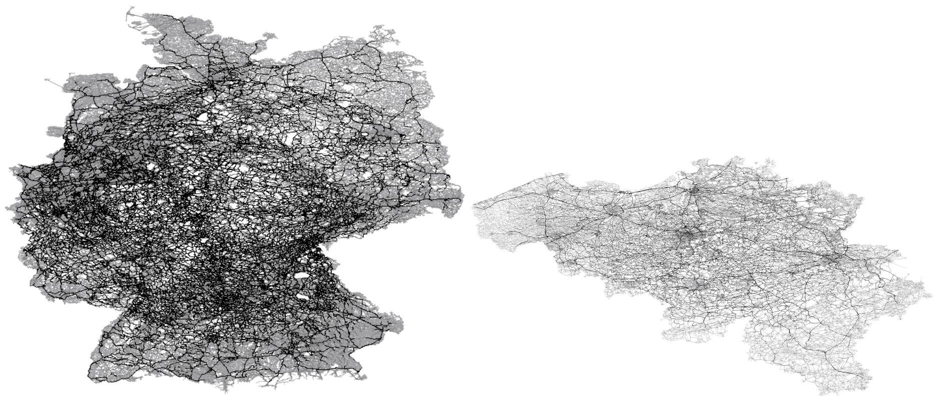


Fig. 6. Germany (left) and Belgium (right) at equilibria states with 10^6 demands each.

Note that the plot sizes do not reflect the actual sizes of either Germany or Belgium. In fact the size of the area of Germany is larger by an order of magnitude. The plot on the left hand side is much darker because the density of the road network is much higher.