Efficient Computation of Multi-Modal Public Transit Traffic Assignments using ULTRA

Jonas Sauer
jonas.sauer2@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Dorothea Wagner
dorothea.wagner@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Tobias Zündorf
zuendorf@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

We study the problem of computing public transit traffic assignments in a multi-modal setting: Given a public transit timetable, an additional unrestricted transfer mode (e.g., walking), and a set of origin-destination pairs, we aim to compute the utilization of all vehicles. While it has been shown that unrestricted walking can significantly improve journeys, computing such journeys efficiently remains algorithmically challenging. Since traffic assignments require the computation of millions of shortest paths, using a multi-modal network has previously not been feasible. In this work we combine the novel ULTRA [2] approach, which enables UnLimited TRAnsfers at the cost of a short preprocessing phase, with a state-of-the-art assignment algorithm, making multi-modal assignments practical. Careful algorithm engineering results in an efficient assignment algorithm, which even outperforms the algorithm it is based on, while enabling unlimited walking for the first time. Finally, we evaluate our algorithm on real worl data, where it computes over 15 million journeys in less than 17 seconds, showing its efficiency.

CCS CONCEPTS

• **Applied computing** → **Transportation**;
• **Theory of computation** → **Shortest paths**;
• **Mathematics of computing** → *Graph algorithms.*

KEYWORDS

Public Transit, Traffic Assignment, Multi-Modal, Algorithm.

ACM Reference Format:
Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. 2019. Efficient Computation of Multi-Modal Public Transit Traffic Assignments using ULTRA. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3347146.3359354

SUPPLEMENTAL MATERIAL

A full version is available at https://arxiv.org/abs/1909.08519.

FUNDING

This research was funded by the DFG under grant no.: WA 654123-2.

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3359354

1 INTRODUCTION

Traffic assignments are an important tool for planning and analyzing transportation networks. Efficient assignment algorithms allow to predict how new infrastructure could improve traffic flows, or to test the limits of existing networks, based on historic, empiric, or expected passenger demand data. For this, the demand is given as a list of origin-destination pairs, where each pair is associated with a desired departure time. A basic variant of the assignment problem then asks for the expected utilization of each vehicle (i.e., the number of passengers using the vehicle) in the public transit network at each point in time. A more intricate second variant additionally asks for a mapping from the origin-destination pairs onto actual journeys through the network that constitute the overall utilization of the vehicles. Solving either of these problems efficiently requires both a fast route planning algorithm (computing journeys for every origin-destination pair) and sophisticated decision models (reflecting which journeys passengers would choose).

1.1 Related Work

In general, traffic assignment problems can be subdivided into two sub-problems. First, computing traffic assignments requires solving a classical route planning problem: Given the origin, destination, and desired departure time of a passenger, find a set of all journeys that the passenger could use. Second, a discrete choice model is required, which reflects the behavior of passengers in the real world and predicts the likelihood of each journey being used.

Many route planning algorithms specialized for timetable networks have been developed in recent years [1]. Especially notable is CSA [4], which is simple yet efficient. Considering unlimited walking reduces travel times significantly but cannot be done efficiently by most algorithms [9, 13]. However, the novel ULTRA approach efficiently enables unlimited walking for many algorithms [2].

An overview of public transit traffic assignment models, algorithms, and decision models can be found in [8, 12]. One approach are sequential route choice models [6], which fit well with CSA [3].

1.2 Our Contribution

In this work we present a novel public transit traffic assignment algorithm that considers not only the timetable network but also unlimited walking. For our new algorithm we combine the ULTRA preprocessing with the CSA-based assignment algorithm presented in [3]. For this, we show how ULTRA can be extended to one-to-many queries, which are required in the context of assignments. Furthermore we present a new grouping technique that increases the accuracy of the result and reduces computation time.

2 PRELIMINARIES

In this section we introduce the notation used throughout the paper and the algorithms upon which we build.

2.1 Public Transit Routing

We model a public transit network as a 4-tuple $(\mathcal{S}, \mathcal{C}, \mathcal{T}, G)$ consisting of a set of *stops* $\mathcal{S}$, a set of *connections* $\mathcal{C}$, a set of *trips* $\mathcal{T}$, and a directed, weighted *walking graph* $G = (\mathcal{V}, \mathcal{E})$. Each *connection* $c \in \mathcal{C}$ is a 5-tuple $(v_{\text{dep}}(c), v_{\text{arr}}(c), \tau_{\text{dep}}(c), \tau_{\text{arr}}(c), \text{trip}(c))$. It represents a vehicle driving from a *departure stop* $v_{\text{dep}}(c) \in \mathcal{S}$ to an arrival stop $v_{\text{arr}}(c) \in \mathcal{S}$ without halting in-between, departing at the *departure time* $\tau_{\text{dep}}(c)$ and arriving at the *arrival time* $\tau_{\text{arr}}(c)$. A *trip* $\text{tr} = (c_1, \ldots, c_k) \in \mathcal{T}$ is a sequence of connections served consecutively by a vehicle. Every connection is part of exactly one trip, which is indicated by $\text{trip}(c)$. Each stop $v \in \mathcal{S}$ is associated with a *departure buffer time* $\tau_{\text{buf}}(v)$. When arriving at v, passengers must observe the departure buffer time $\tau_{\text{buf}}(v)$ before they can enter a connection departing at v, unless they arrived via the same trip.

The walking graph $G = (\mathcal{V}, \mathcal{E})$ consists of a set of *vertices* $\mathcal{V}$ with $\mathcal{S} \subseteq \mathcal{V}$ and a set of *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each edge $(v, w) \in \mathcal{E}$ has a *walking time* $\tau_{\text{walk}}(v, w) \in \mathbb{N}_0$ which is the time needed to traverse the edge. The walking time of a path in G is defined accordingly. We denote the walking time of the shortest path in G from v to w by $\text{dist}(v, w)$. Unlike in restricted walking scenarios, G does not have to be transitively closed or fulfill the triangle inequality.

We call the movement of a passenger from an origin vertex to a destination vertex through the public transit network a *journey*. Formally a journey is an alternating sequence of the vehicles used by the passenger and paths in the walking graph that allow for transferring between these vehicles. By $\tau_{\text{arr}}(J)$ we denote the arrival time of the journey J at its destination d. The *perceived arrival time* (PAT) $\tau^p_{\text{arr}}(J)$ of a journey J is the sum of the actual arrival time $\tau_{\text{arr}}(J)$ and several penalties for inconveniences during the journey: A transfer penalty $\lambda_{\text{trans}} \in \mathbb{R}^+_0$ is added for each transfer between two vehicles. Time spent walking or waiting for a vehicle is multiplied by the walking penalty $\lambda_{\text{walk}} \in \mathbb{R}^+_0$ or waiting penalty $\lambda_{\text{wait}} \in \mathbb{R}^+_0$, respectively. Additionally, the PAT accounts for delays by incorporating alternative choices for delayed connections. The precise formal definition of the PAT is given in [3].

A *v-w-profile* $f^{v,w}(\tau)$ between two vertices $v, w \in \mathcal{V}$ is a function that maps each departure time τ to the minimal PAT among all v-w-journeys that depart at v no earlier than τ. If no feasible journey exists, we define $f^{v,w}(\tau)$ as ∞. The profile can be represented as a piecewise linear function where each segment represents one journey, as well as an additional value τ^p_{walk} for the shortest pure walking journey. Evaluating a profile at a time τ is done by returning the minimum of τ^p_{walk} and the value of the function at τ.

2.2 Problem Statement

A traffic assignment problem takes as input a public transit network and a *demand* $\mathcal{D}$, which is a set of origin-destination pairs. Each origin-destination pair $p = (o, d) \in \mathcal{D}$ represents a passenger who wants to travel from the *origin vertex* $o \in \mathcal{V}$ to the *destination vertex* $d \in \mathcal{V}$, starting at a desired departure time $\tau_{\text{dep}}(p)$. The objective is to assign each origin-destination pair $p = (o, d) \in \mathcal{D}$ to a probability space consisting of journeys that depart at o no earlier than $\tau_{\text{dep}}(p)$ and end at d. The probability of a journey should reflect the likelihood of a real passenger using the journey. Summing the probabilities of all journeys containing a connection c yields the *utilization* $u(c)$, which is the expected number of passengers using c.

2.3 CSA-Based Assignment

The CSA-based assignment algorithm from [3] computes a traffic assignment by simulating the movement of individual passengers through the public transit network. All origin-destination pairs with the same destination vertex d are processed simultaneously. Pairs with different destinations can be processed in parallel. The assignment for all pairs belonging to one destination is done in three phases: (1) the PAT profile computation, (2) the actual assignment phase, and (3) an optional cycle removal phase.

The first phase computes a partial PAT profile $f^{v,d}_{\text{wait}}$ for each vertex $v \in \mathcal{V}$, where $f^{v,d}_{\text{wait}}(\tau)$ is the minimum PAT for a passenger waiting at v for the best vehicle departing directly at v after τ. Additionally, it computes three PATs for every connection, which are later used to decide if passengers use the connection on their journey to the destination. These values are computed with a single scan of the connection array in decreasing order of departure time.

The second phase uses these PATs to compute journeys for the origin-destination pairs. This is done by simulating the movement of individual passengers through the network and recording the journey they take. The *passenger multiplier* $\lambda_{\text{mul}} \in \mathbb{N}$ controls how many passengers are generated for each origin-destination pair $p = (o, d)$ at o. The passengers are routed through the network by processing the connections in increasing order of departure time. For each connection c, the algorithm makes three decisions: (1) whether passengers waiting at $v_{\text{dep}}(c)$ enter c, (2) whether passengers using c disembark at $v_{\text{arr}}(c)$, and (3) whether disembarking passengers walk to another stop or keep waiting at $v_{\text{arr}}(c)$. In all three cases, the PATs of all available options were computed in the first phase. Given a *choice set* of k options and their respective PATs, the *gain* of the i-th option is defined as the PAT difference to the best other option, plus a *delay tolerance* $\lambda_{\Delta\max} \in \mathbb{R}^+_0$. If the PAT difference is more than $\lambda_{\Delta\max}$, the gain is set to 0 and the option is eliminated from the choice set. A decision is then made for each passenger by choosing an option randomly, with a probability proportional to its gain. Doing this for each connection results in the passengers gradually moving towards the destination.

2.4 ULTRA

The ULTRA approach [2] is based on the observation that long walking paths between vehicles are only rarely required, while long walking paths as the first or last leg of a journey occur quite frequently. ULTRA exploits this fact by precomputing a shortcut graph $G' = (\mathcal{S}, \mathcal{E}')$ containing all necessary inter-vehicle transfers. Public transit algorithms can then find transfers between trips by scanning these shortcuts instead of the (much larger) unrestricted transfer graph. Initial and final walking paths of a journey can be computed efficiently with a specialized one-to-many shortest path algorithm called Bucket-CH [5, 7]. This approach is only viable for solving one-to-one queries in the public transit network. However, as the CSA-based assignment algorithm is based on an all-to-one CSA , integrating ULTRA is not straightforward.

3 ULTRA-ASSIGNMENT ALGORITHM

In this section we give an overview of our ULTRA-Assignment algorithm, focusing on the differences to the original assignment algorithm introduced in [3]. Pseudocode is given in Algorithm 1.

Handling Departure Buffer Times. In the original assignment algorithm, the departure buffer time was considered part of the waiting time and therefore the waiting penalty λ_{wait} was applied to it. However, since the departure buffer time must always be observed, it may not be desirable to penalize it to the same degree as waiting, or at all. Therefore we introduce a new buffer time penalty $\lambda_{\text{buf}} \in \mathbb{R}_0^+$ that may be different from the waiting penalty λ_{wait}.

Unrestricted Walking Using ULTRA. The original assignment algorithm used CSA variants with restricted walking in both the PAT computation phase and the assignment phase. We extend the algorithm to unrestricted walking by replacing CSA with ULTRA-CSA in both phases. This is straightforward for the intermediate and final transfers, but not for the initial transfers because each origin that occurs in the demand is a potential source vertex. To handle initial transfers, we perform a Bucket-CH from each origin vertex (line 4) and store the distances to all stops in a list of stop-distance tuples, sorted in ascending order of distance (line 5). The PAT computation phase then only computes profiles at each stop, excluding initial transfers. Whenever passengers are generated for an origin-destination pair $p = (o, d)$, we evaluate the initial transfers by iterating over the stop-distance tuples (line 14), computing the corresponding PATs and adding them to the choice set. Once the departure time plus the perceived walking time τ^p_{walk} exceeds the best PAT found so by far by more than $\lambda_{\Delta\max}$, we can stop iterating through the list since all further options will have a probability of 0 (line 18). The remainder of the assignment phase continues as in the original assignment algorithm. Refer to the full version of this paper [10] for further details.

Grouping Multiplied Passengers. The original CSA-based assignment algorithm generated λ_{mul} copies of each passenger in the demand and then simulated the movement of all these copies independently. This approach leads to redundant work because different copies of the same passenger will often make the same choices. We solve this problem by grouping passengers that make the same choices together into *passenger groups*. The number of passengers in a group is indicated by the *group size* γ. At the start of the assignment phase, we generate one group of size λ_{mul} for each origin-destination pair. When distributing a group of size γ across k options with probabilities $P_1, \ldots, P_k$, we split it into k groups of sizes $\lfloor \gamma P_1 \rfloor, \ldots, \lfloor \gamma P_k \rfloor$ and route each group according to the corresponding option. Because the group sizes are rounded down, some of the original γ passengers may still be left over afterwards. These passengers are handled individually by randomly choosing an option for each passenger according to the probabilities and adding the passenger to the corresponding group, as in the original assignment algorithm. If the probability of an option is lower than $1/\gamma$ and none of the leftover passengers are assigned to it, the corresponding group becomes empty and is deleted. In addition to improving the performance, this grouped approach increases the accuracy of the result because the resulting group sizes match the expected distribution of the passengers as closely as possible.

Algorithm 1: ULTRA-Assignment.

Input: Public transit network $(\mathcal{S}, C, \mathcal{T}, G = (\mathcal{V}, \mathcal{E}))$, shortcut graph $G' = (\mathcal{S}, \mathcal{E}')$, and demand $\mathcal{D}$
Output: Utility $u\colon C \to \mathbb{R}_0^+$ and corresponding journeys $\mathcal{J}$

1 Let O be the set of all origins with demand in $\mathcal{D}$
2 Let D be the set of all destinations with demand in $\mathcal{D}$
3 **for each** $o \in O$ **do**
4 $\quad N(o) \leftarrow \{(v, \text{dist}(o, v)) \mid v \in \mathcal{S}\}$ `// Using Bucket-CH`
5 $\quad$ Sort $N(o)$ in ascending order of distance $\text{dist}(o, \cdot)$
6 Sort $\mathcal{D}$ lexicographically by destination, origin
7 Sort C ascending by departure time
8 **for each** destination $d \in D$ **do**
9 $\quad$ Compute PAT profiles from every stop to d
10 $\quad$ **for each** $p = (o, d) \in \mathcal{D}$ *with destination* d **do**
11 $\quad\quad$ Generate passenger group g of size λ_{mul} for p
12 $\quad\quad$ $\mathcal{J} \leftarrow \mathcal{J} \cup \{J_g = \{\}\}$
13 $\quad\quad$ Let C be an empty choice set for p
14 $\quad\quad$ **for each** $(v, \text{dist}(o, v)) \in N(o)$ **do**
15 $\quad\quad\quad$ $\tau_{\text{dep}} \leftarrow \tau_{\text{dep}}(p) + \text{dist}(o, v)$
16 $\quad\quad\quad$ $\tau^p_{\text{walk}} \leftarrow \lambda_{\text{walk}} \cdot \text{dist}(o, v)$
17 $\quad\quad\quad$ $\bar{\tau}^p_{\text{arr}} \leftarrow \min\{\tau^p_{\text{arr}} \mid (\tau^p_{\text{arr}}, \cdot, \cdot) \in C\} + \lambda_{\Delta\max}$
18 $\quad\quad\quad$ **if** $\tau_{dep} + \tau^p_{walk} > \bar{\tau}^p_{arr}$ **then break**
19 $\quad\quad\quad$ $\tau^p_{\text{arr}} \leftarrow f^{v,d}_{\text{wait}}(\tau_{\text{dep}} + \tau_{\text{buf}}(v)) + \tau^p_{\text{walk}} + \lambda_{\text{buf}} \cdot \tau_{\text{buf}}(v)$
20 $\quad\quad\quad$ $C \leftarrow C \cup \{(\tau^p_{\text{arr}}, v, \tau_{\text{dep}})\}$
21 $\quad\quad$ Evaluate which choice from C the passengers use
22 $\quad$ **for each** $c \in C$ **do**
23 $\quad\quad$ Evaluate if passengers waiting at $v_{\text{dep}}(c)$ enter c
24 $\quad\quad$ $u(c) \leftarrow$ Number of passengers in c
25 $\quad\quad$ Add c to journeys J_g of groups g in c
26 $\quad\quad$ Evaluate if passengers using c disembark at $v_{\text{arr}}(c)$
27 $\quad\quad$ Evaluate if passengers at $v_{\text{arr}}(c)$ can transfer to d
28 $\quad\quad$ Evaluate to which stop passengers at $v_{\text{arr}}(c)$ transfer
29 **for each** $J_g \in \mathcal{J}$ **do** Remove cycles from J_g

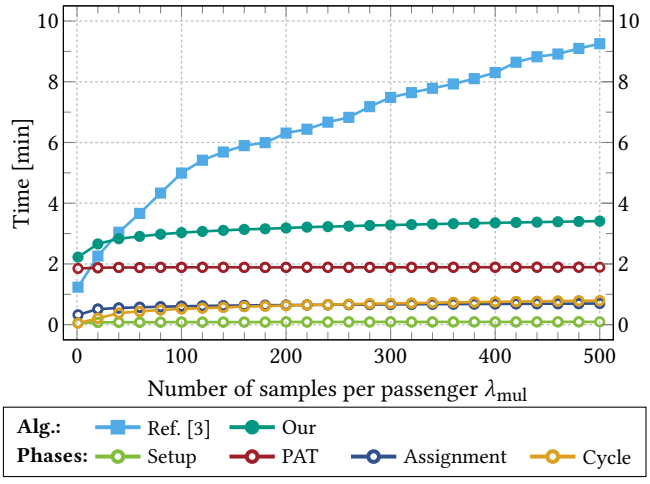

Figure 1: Sequential execution time of ULTRA-Assignment and its sub-phases compared to the algorithm from [3], depending on the passenger multiplier (average of 10 runs).

4 EXPERIMENTS

All algorithms were implemented in C++17 and compiled with GCC version 7.3.1 and optimization flag -O3. We conducted all experiments on a machine with two 8-core Intel Xeon Skylake SP Gold 6144 CPUs clocked at 3.5 GHz with 192 GiB of DDR4-2666 RAM.

For our experiments we use public transit data from the greater region of Stuttgart, Germany. The network contains 13 941 stops and 3 710 524 connections; the demand data comprises 1 249 910 origin-destination pairs. This benchmark instance was previously used in [3, 11]. For the unlimited walking we added a footpath graph with 1 170 198 vertices and 3 710 524 edges from OpenStreetMap[1], assuming a walking speed of 4.0 km/h.

4.1 Preprocessing

Our assignment algorithm requires preprocessing for the Bucket-CH queries and the ULTRA shortcuts, which in turn requires a core graph. The CH was computed in 2:44 min (1 thread) and added 5 469 298 shortcuts. The core graph was computed using CH contractions until an average vertex degree of 16 was reached, which took 2:30 min (1 thread) and yielded a core graph with 25 477 vertices and 407 664 edges. Finally, the ULTRA preprocessing took another 2:03 min (16 threads) and resulted in 74 038 shortcuts.

4.2 ULTRA-Assignment

Using a passenger multiplier λ_{mul} of 100, the assignment computation takes 181.9 seconds sequentially and 16.8 seconds in parallel with 16 threads. This is more than a factor of 2 faster than the CSA-based assignment with limited walking, which has a running time of 36.9 seconds on the same instance. Qualitative figures like the average travel time or number of used trips are only slightly lower (less than 5%) when comparing unlimited to limited walking. However, the number of distinct journeys per origin-destination pair increases from 9.27 to 12.79 when switching from limited walking to unlimited walking. Enabling unlimited walking also increases the number of origin-destination pairs for which feasible journeys exist from 1 209 761 to 1 246 337.

The most important tuning parameter of our assignment algorithm is the passenger multiplier λ_{mul}, which determines the number of samples evaluated in our Monte Carlod method. As such, it has a significant impact on both the accuracy of the result and the computation time. While the impact that λ_{mul} has on the accuracy is quite clear (the logarithm of λ_{mul} corresponds to the number of decimal places to which the result is computed), its impact on the running time is not at obvious. We therefore evaluate the performance of our algorithm depending on the passenger multiplier in Figure 1, which also includes a curve for the original CSA-based assignment. The running time of both algorithms increases sub-linearly with respect to λ_{mul}. However, the curve of our new algorithm (dark green) grows significantly slower due to our passenger grouping approach. We also report the running times for 4 sub-phases of our algorithm (the colors correspond to the line numbers in Algorithm 1). The plot shows that the most costly phase of our algorithm is the PAT computation phase. This observation matches our expectations, as the PAT computation phase has to scan the complete multi-modal transportation network, including final transfers.

[1] http://download.geofabrik.de/

5 CONCLUSION

We presented a new public transit traffic assignment algorithm that can handle unrestricted walking and is faster than previous approaches that were restricted to the pure public transit network. We achieved this by integrating the novel ULTRA approach for handling unrestricted transfers into a state-of-the-art assignment algorithm. By doing this, we developed the first one-to-many query algorithm that is able to use ULTRA shortcuts. We proceeded with improving the overall performance of the assignment algorithm, so that we can compute an assignment for over 1.2 million origin-destination pairs in less than 17 seconds. In a thorough experimental study we demonstrated the validity of our approach. In particular we showed that our algorithm solves the assignment problem efficiently, regardless of the applied discrete choice model, the demand data, or the requested accuracy.

For future work, we would like to improve the overall quality of the computed assignments by integrating more complex journey choice models. More sophisticated models could for example consider vehicle capacities and reduce the likelihood of assigning passengers to overcrowded vehicles. Furthermore, it would be interesting to correlate the probabilities of journeys that overlap partially, for example if both use the same vehicle as a leg of the journey.

REFERENCES

[1] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. 2016. Route Planning in Transportation Networks. In *Algorithm engineering*. Springer, 19–80.

[2] Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. 2019. UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution. In *27th Annual European Symposium on Algorithms (ESA 2019) (Leibniz International Proceedings in Informatics (LIPIcs))*, Michael A. Bender, Ola Svensson, and Grzegorz Herman (Eds.), Vol. 144. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 14:1–14:16. https://doi.org/10.4230/LIPIcs.ESA.2019.14

[3] Lars Briem, Sebastian Buck, Holger Ebhart, Nicolai Mallig, Ben Strasser, Peter Vortisch, Dorothea Wagner, and Tobias Zündorf. 2017. Efficient Traffic Assignment for Public Transit Networks. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 75. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[4] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. 2013. Intriguingly Simple and Fast Transit Routing. In *International Symposium on Experimental Algorithms*. Springer, 43–54.

[5] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08) (Lecture Notes in Computer Science)*, Vol. 5038. Springer, 319–333.

[6] Guido Gentile and Andrea Papola. 2006. An Alternative Approach to Route Choice Simulation: The Sequential Models. In *Proceedings of the European Transport Conference*.

[7] Sebastian Knopp, Peter Sanders, Dominik Schultes, Frank Schulz, and Dorothea Wagner. 2007. Computing Many-to-Many Shortest Paths Using Highway Hierarchies. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*. SIAM, 36–45.

[8] Michael Patriksson. 2015. *The Traffic Assignment Problem: Models and Methods*. Courier Dover Publications.

[9] Duc-Minh Phan and Laurent Viennot. 2019. Fast Public Transit Routing with Unrestricted Walking through Hub Labeling.

[10] Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. 2019. Efficient Computation of Multi-Modal Public Transit Traffic Assignments using ULTRA. arXiv:arXiv:1909.08519

[11] Johannes Schlaich, Udo Heidl, and Regine Pohlner. 2011. Verkehrsmodellierung für die Region Stuttgart – Schlussbericht. (2011).

[12] Kenneth E Train. 2009. *Discrete Choice Methods with Simulation*. Cambridge university press.

[13] Dorothea Wagner and Tobias Zündorf. 2017. Public Transit Routing with Unrestricted Walking. In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.