Extending Accessibility Analysis With True Multi-Modality

Master Thesis

Author: Moritz Gottschling (Student ID: 7350270)
Supervisor: Univ.-Prof. Dr. Wolfgang Ketter
Co-Supervisor: Philipp Peter

Department of Information Systems for Sustainable Society
Faculty of Management, Economics and Social Sciences
University of Cologne

September 15, 2023

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs. 1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Moritz Gottschling

Köln, den xx.xx.20xx

Abstract

[Abstract goes here (max. 1 page)]

Contents

List of Figures

List of Tables

1 Introduction

"Results show that bicycles significantly reduce the average transfer times, the average path length of passengers' trips and the Gini coefficient of an urban public transport network" (Yang et al., 2018).

Public transport frequency is significantly positively correlated with the number of bicycle trips, especially short and medium distance trips up to 3 km (Radzimski & Dzięcielski, 2021).

(Murphy & Usher, 2015) conduct a questionnaire, which shows that 39% of bicycle sharing users (in Dublin) use bicycle sharing in conjunction with another mode of transport. Of those, 91.5% use public transport, which indicates that bicycle sharing is synergetic with public transport.

(Fishman, Washington, & Haworth, 2013) perform a literature review on bicycle sharing in general and find that bicycle sharing is synergetic with public transport.

(Ma, Liu, & Erdoğan, 2015) run a linear regression with data ... in which the number of passengers of public transport is regressed on the number of bicycle sharing trips. They find a positive correlation between the two and conclude that bicycle sharing and public transport are complementary. As a possible reason for this, they state that bicycle sharing can be used to solve the first and last mile problem.

2 Related Work

...

2.1 Accessibility Analysis

... In order to assess the accessibility from a given origin to one or multiple points of interest, a routing algorithm is required. The routing algorithm finds the shortest path from the origin to the destination.

2.2 Routing Algorithms

The primary goal of routing algorithms is to identify the optimal path between a designated origin and a specific destination. Typically, this is captured using a graph representation:

$$G = (V, E)$$

where V represents a set of nodes (or locations) and E encapsulates the set of edges, which correspond to connections between these nodes.

For each edge $e \in E$, there's an associated weight $w(e) \in \mathbb{R}$ that characterizes the cost of traversing it. This cost might be determined by factors such as distance or travel time. Consequently, the shortest path can be expressed as:

$$\langle v_0, e_0, v_1, e_1, \ldots, v_n \rangle$$

Here, v_0 denotes the origin, v_n the destination, and the edges must connect the nodes in the sequence:

$$e_i = (v_i, v_{i+1}) \quad \text{for} \quad i \in \{0, \ldots, n-1\}$$

In accessibility contexts, the primary concern frequently revolves around determining the accumulated cost, $d(v_n)$, to reach the destination rather than the actual path.

In more complex real-world scenarios, the problem often encompasses multiple objectives, such as considering both time and monetary cost of travel. Under these circumstances, the edge weight is represented as a vector:

$$w(e) \in \mathbb{R}^k$$

where k stands for the total objectives count. Unlike the simpler single-objective case with a singular optimal path, the multi-objective scenario yields a Pareto set, constituting several optimal routes.

The value of these paths is depicted using a label:

$$l \in \mathbb{R}^k$$

where $l_i \in \mathbb{R}$ denotes the value for the i-th objective. This label can be thought of as a multidimensional extension of $d(v_n)$ from the single-objective scenario. The Pareto set associated with destination node v_n is often termed as a bag, expressed as $B(v_n)$, comprising labels that are not dominated by each other. Domination is defined as follows: l' dominates l if $l'_i \leq l_i$ for all $i \in \{1, \ldots, k\}$ and $l'_i < l_i$ for at least one $i \in \{1, \ldots, k\}$. Intuitively, this means that l' is at least as good as l in all objectives and strictly better in at least one objective.

The goal of routing algorithms used in accessibility analysis is finding the distance in the single objective case and the bag in the multi objective case. For accessibility analysis routing algorithms are often altered to not find the optimal path(s) between two nodes, referred to as one-to-one query, but the path from a single origin to all other nodes in the network, which we call one-to-all query.

2.2.1 Dijkstra

The most straightforward approach to compute the shortest paths in a graph is the Dijkstra algorithm (Dijkstra, 1959).

Dijkstra's algorithm initiates at a designated start node $s \in V$ and employs a priority queue to systematically determine the shortest path to each subsequent node $v \in V$. Initially, the distance to the start node s is set to zero, while the distances to all other nodes are set to infinity. In each iteration, the algorithm dequeues the node u with the smallest known distance from the priority queue. It then examines each outgoing edge $e = (u, v)$ from u, updating the distance to v if a shorter path through u is discovered. Specifically, if $\text{dist}(u) + w(e) < \text{dist}(v)$, then $\text{dist}(v)$ is updated to $\text{dist}(u) + w(e)$, and v is enqueued into the priority queue for future exploration. The node u is marked as visited by adding it to the set V_{visited}. Depending on the goal, the algorithm terminates either when the destination node is dequeued (one-to-one) or when the priority queue is empty (one-to-all).

However, this simple approach has multiple problems. Firstly, the Dijkstra algorithm is not able to handle multiple criteria. Secondly, the runtime of Dijkstra's algorithm is $O(|E| + |V| \log |V|)$, which is too slow for large graphs.

2.2.2 MLC

The Multi-Label-Correcting (MLC) (Hansen, 1980) algorithm is an extension of Dijkstra's algorithm to handle multi-objective scenarios. As mentioned in Section

2.2 in the multi-objective case we try to find the bag of the destination node. Specifically, for k criteria, each node v retains a bag of k-dimensional labels. Such a list encapsulates a set of Pareto-optimal paths from the starting node to v. Similarly to Dijkstra's algorithm, MLC initializes all nodes with an empty bag, except for the start node, which is initialized with a label of $(0, \ldots, 0) \in \mathbb{R}^k$. Each iteration extracts the lexicographically smallest label, as opposed to selecting the node with the minimum distance. When a label is extracted and v is its corresponding node, updates are made for all connected edges (v, w). The update process consists of comparing a newly generated tentative label against all labels within the bag of w. This new label is only inserted into the bag if it isn't dominated by any existing label. Conversely, any label now dominated by the new entry is removed. Each time a label is inserted into a bag, it is also inserted into the priority queue. The algorithm terminates when the priority queue is empty.

The major drawback of the MLC algorithm is its runtime, which is even slower than Dijkstra's algorithm, because each node can be visited multiple times.

2.2.3 Graph-based Algorithms in Public Transport

In the context of accessibility analysis the previously mentioned algorithms can be used directly for walking, cycling and driving networks. However, public transport networks pose a challenge, since they contain time-dependent information, such as the departure time of a trip. To overcome this challenge two different approaches are commonly used, the time-expanded and the time-dependent approach, as explained by (Müller-Hannemann, Schulz, Wagner, & Zaroliagis, 2007). While enabling the use of graph-based algorithms, both approaches still suffer from the previously mentioned runtime problems Dijkstra's algorithm and MLC have.

2.2.4 RAPTOR

To overcome the runtime problems of graph-based approaches, (Delling, Pajor, & Werneck, 2015) introduce one of the most prominent routing algorithms for public transport, called Round based Public Transit Optimized Router algorithm (RAPTOR). Unlike traditional Dijkstra-based algorithms, RAPTOR operates in rounds, looking at each route (such as a bus line) in the network at most once per round.

As RAPTOR does not operate on a graph, we first introduce the problem statement. Raptor operates on a scheduled network consisting of routes r, trips t, stops p, and stop times that associate trips with stops. A route is associated with a sequence of stops $\mathit{stops}(r) = \langle p_1, \ldots, p_n \rangle$. A route has multiple trips ordered

by their departure time $trips(r) = \langle t_1, \ldots, t_m \rangle$. One trip associates arrival and departure times with each stop of the route, denoted by $arrivalTime(t, s) \in \mathbb{N}$ and $departureTime(t, s) \in \mathbb{N}$ respectively. Trips of the same must not overtake each other, formally:

$$departureTime(t_i, p_j) \leq arrivalTime(t_{i+1}, p_j)$$

for all $i \in \{1, \ldots, m-1\}$ and $j \in \{1, \ldots, n\}$. Each stop p has a minimal exchange time $\tau_{ch}(p) \in N$ associated with it. Often, the exchange time is set to a fixed time $\tau_{ch}(p) = \tau_{ch}$ for all stops p. When transferring from a trip t to another trip t' within at a stop p, the exchange time has to be smaller than the difference in arrival and departure time of the two trips, formally:

$$arrivalTime(t, p) + \tau_{ch}(p) \leq departureTime(t', p)$$

In addition to transfer within stops, RAPTOR also allows footpaths. Footpaths allow transferring from one stop to another without using public transport, therefore, they are time-independent. Each footpath is associated with a travel time $l(p, p')$. The input of the RAPTOR algorithm, in addition to the previously described scheduled network, are source stop p_s, and, in the case of a one-to-one query, target stop p_t, as well as, the departure time at the source stop τ.

RAPTOR operates in rounds. Before the first round, some variables are initialized. We denote the earliest possible arrival time at iteration i with $\tau_i(p)$ and the best earliest possible arrival time over the course of all iterations with $\tau^\star(p)$. For the source stop, τ_p, we set $\tau_0(p) = \tau$ and $\tau^\star(p) = \tau$. For all other stops, we set $\tau_0 = \infty$ and $\tau^\star = \infty$. In addition, we initialize a set of marked nodes M to only contain the source stop p_s and a set of marked route-stop pairs, denoted by Q, to the empty set. A route-stop pair is simply a tuple that contains a route and one of its stops. The set of marked stops will contain all stops whose earliest possible arrival time has been updated in the current round. Similarly, the set of marked route-stop pairs contains the routes of the marked stops, together with the earliest stop of that route that has been marked.

Each round consists of three major steps. In the first step, the routes that have to be iterated are collected. In the second step, the routes are iterated by "hopping" on their trips. And in the third stage, potential footpaths are explored.

First, we clear the set of marked route-stop pairs Q. Then we check the routes that are connected to each marked stop. For each of these routes, we store the route-stop pair in Q. However, the routes in Q should be unique. If there are two marked stops that are connected to the same route, we choose the stop that is earlier in the sequence of stops of that route. Now, we clear the set of marked

stops.

We iterate the route-stop pairs in Q. The following step can be regarded as hopping on the earliest possible trip that we can catch of that route at that stop. For each route-stop (r, p) pair, we iterate over the stops in r in the sequence that is associated with r, beginning with p. We check for the earliest possible trip that we can catch regarding the last arrival time at the current stop $\tau_{k-1}(p)$ and the minimum exchange time $\tau_{ch}(p)$. If there is a trip that is possible to catch, we save it as the current trip t_{curr} and continue to iterate the stops of the route r. Now that we are on a trip, we have to check whether we need to update the earliest possible arrival time of the current stop $\tau_k(p)$ and $\tau^\star(p)$ by comparing the stop time of the current trip with the best earliest arrival time of that stop $\tau^\star(p)$, formally:

$$\tau_k(p) = \min\{\tau_k(p), arrivalTime(t_{curr}, p)\}$$

Here one optimization comes into play. In the case an update is necessary, we also add the current stop p to the marked stops.

Lastly, we check all marked stops for potential footpaths. Remember: the marked stops are those for which the earliest possible arrival time was updated in this iteration. For each footpath that is connected to a marked stop, we check whether the earliest possible arrival time of the other stop could be improved by the footpath. If that is the case, we update the earliest arrival times and also mark that stop.

If no stops are marked, then there are no new routes to iterate, and the algorithm stops.

After termination

$$\tau_k(p)$$

contains the earliest possible arrival time at stop p with at most k transfers.

One limitation of RAPTOR is the transfer graph, which is used to represent footpaths. The transfer graph has to be transitively closed, which means that each node has to be connected with an edge to all other nodes that can be reached from that node. This has the advantage that in the algorithm we only have to check for direct neighbors of a stop, which is very fast. In practice, there are many possibilities how the transfer graph could look. First, we should note that a realistic transfer graph should be derived from a street network, as passengers should be able to walk from one stop to another using sidewalks. To keep the transfer graph small, one could limit the maximum walking distance. However, this may remove optimal journeys from the search space. Therefore, finding a fitting transfer graph is challenging.

Through its round-based nature, RAPTOR is able to optimize for two criteria

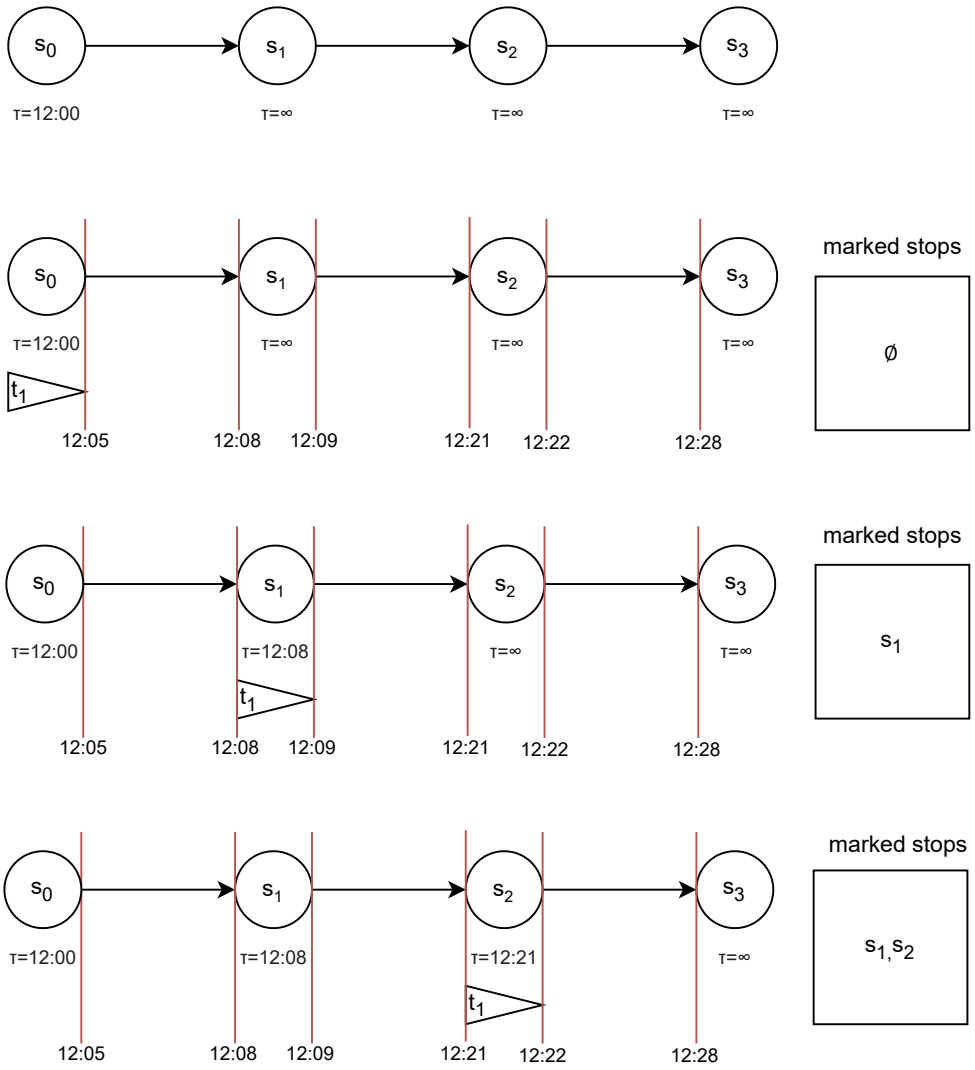

Figure 1: Iterating a route in RAPTOR

at the same time. However, RAPTOR cannot incorporate more criteria and one of the criteria will always be the number of transfers.

2.2.5 McRAPTOR

McRAPTOR (Delling et al., 2015) is an extension of RAPTOR that allows an arbitrary number of criteria. Like MLC, McRAPTOR also uses the notion of bags containing non-dominating labels. McRAPTOR does not pose any restrictions on how the objectives are updated during the algorithm.

The algorithm of McRAPTOR only requires slight modifications to the algorithm of RAPTOR. In the initialization step, each stop p is assigned an empty bag, except the source stop p_s, which is assigned a bag containing a starting label. The starting label can be defined as an input, but is usually $(\tau, 0, 0, \ldots, 0)$, where τ is the departure time at the source stop.

When iterating over the route-stop pairs $(r, p$, McRAPTOR creates a route bag that contains all labels that are in the current bag of p. In addition labels in the route bag are associated with a trip. During creation of the route bag, each label in the route bag is associated with the first trip that is possible to catch according to the labels earliest arrival time at the current stop p. Then the route is processed, stop by stop, just like in RAPTOR. At each stop the labels in the route bag are updated according to the current trip. This update must include updating the earliest arrival time, but can also include updates to other criteria. After the route has been processed, the route bag is merged into the bag of the current stop. Merging a bag B_1 into a bag B_2 means that all labels in B_1 that are not dominated by any label in B_2 are added to B_2 and all labels in B_2 that are dominated by a label in B_1 are removed from B_2. After the route bag has been merged into the bag of the current stop, the bag of the current stop is merged into the route bag. Lastly, the trips that are associated with the labels in the route bag are updated according to the labels earliest arrival time at the current stop.

Each time a label is added to a stop bag, this stop is marked. If no stop is marked after a round, the algorithm terminates.

Note that McRAPTOR allows updates to the route bags at any time during processing. When and how the route bag should be updated fully depends on the objective and what it represents.

While McRAPTOR has a slower runtime than RAPTOR it is still magnitudes faster than MLC. However, McRAPTOR still suffers from the same problem as RAPTOR, namely that the transfer graph is hard to compute.

2.2.6 MCR

To overcome problem of RAPTOR (Delling, Dibbelt, Pajor, Wagner, & Werneck, 2013) introduce Multimodal Multicriteria RAPTOR (MCR). MCR modifies McRAPTOR so that the transfer graph must not be transitively closed.

This allows us to use the street network as the transfer graph, which has the benefit that during the traversal of the tranfer graph the objectives can be updated. This is important if we multiple modes of transfer, that contain free-floating vehicle sharing systems. For example, consider the following case. For an optimal journey a passenger has to first walk five minutes to a free-floating bicycle, with which the passenger then travels to the next stop. There is no way to represent this in RAPTOR, because the specifics of the transfer depend on the current label, which is unknown before running the algorithm. Therefore it is not possible to precompute the transfer graph.

MCR is very similar to RAPTOR. It only replaces the footpath traversal step through MLC.

2.2.7 ULTRA

A Appendix

...

References

Delling, D., Dibbelt, J., Pajor, T., Wagner, D., & Werneck, R. F. (2013). Computing Multimodal Journeys in Practice. In V. Bonifaci, C. Demetrescu, & A. Marchetti-Spaccamela (Eds.), *Experimental Algorithms* (pp. 260–271). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-38527-8_24

Delling, D., Pajor, T., & Werneck, R. F. (2015, August). Round-Based Public Transit Routing. *Transportation Science*, *49*(3), 591–604. (implemented in https://transnetlab.github.io/transit-routing/html/index.html

introduction of raptor) doi: 10.1287/trsc.2014.0534

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271.

Fishman, E., Washington, S., & Haworth, N. (2013, March). Bike Share: A Synthesis of the Literature. *Transport Reviews*, *33*(2), 148–165. doi: 10.1080/01441647.2013.775612

Hansen, P. (1980). Bicriterion Path Problems. In G. Fandel & T. Gal (Eds.), *Multiple Criteria Decision Making Theory and Application* (pp. 109–127). Berlin, Heidelberg: Springer. (MCL cited as first apperacne) doi: 10.1007/978-3-642-48782-8_9

Ma, T., Liu, C., & Erdoğan, S. (2015, January). Bicycle Sharing and Public Transit: Does Capital Bikeshare Affect Metrorail Ridership in Washington, D.C.? *Transportation Research Record*, *2534*(1), 1–9. doi: 10.3141/2534-01

Müller-Hannemann, M., Schulz, F., Wagner, D., & Zaroliagis, C. (2007). Timetable Information: Models and Algorithms. In F. Geraets, L. Kroon, A. Schoebel, D. Wagner, & C. D. Zaroliagis (Eds.), *Algorithmic Methods for Railway Optimization* (pp. 67–90). Berlin, Heidelberg: Springer. (MLC

should contain MLC

MLC is referred to as MOSP

MOSP is the origin) doi: 10.1007/978-3-540-74247-0_3

Murphy, E., & Usher, J. (2015, February). The Role of Bicycle-sharing in the City: Analysis of the Irish Experience. *International Journal of Sustainable*

Transportation, *9*(2), 116–125. (questionnaire reports that 39% of bicycle sharing users use it in conjunction with another mode of transport. 91.% of those use it together with public transport) doi: 10.1080/15568318.2012.748855

Radzimski, A., & Dzięcielski, M. (2021, March). Exploring the relationship between bike-sharing and public transport in Poznań, Poland. *Transportation Research Part A: Policy and Practice*, *145*, 189–202. (stops positively correlate with bicycle trips) doi: 10.1016/j.tra.2021.01.003

Yang, X.-H., Cheng, Z., Chen, G., Wang, L., Ruan, Z.-Y., & Zheng, Y.-J. (2018, January). The impact of a public bicycle-sharing system on urban public transport networks. *Transportation Research Part A: Policy and Practice*, *107*, 246–256. (bike sharing can significantly reduce the average transfer times) doi: 10.1016/j.tra.2017.10.017