University of West Bohemia
Faculty of Applied Sciences

ALGORITHMS IN TRANSPORTATION

František Kolovský

doctoral thesis
submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Geomatics

Supervisor: prof. Dr. Ing. Ivana Kolingerová
Department: Department of Geomatics

Pilsen 2021

Západoceská univerzita v Plzni
Fakulta aplikovaných věd

ALGORITMY V DOPRAVĚ

František Kolovský

Disertační práce
k získání akademického titulu doktor
v oboru Geomatika

Školitelka: prof. Dr. Ing. Ivana Kolingerová
Katedra: Katedra geomatiky

Plzeň 2021

Declaration

I declare that this thesis is my original work of authorship that I have created myself. All resources, sources and literature, which I used in my thesis, are cited indicating the full link to the appropriate source.

In Pilsen ....
František Kolovský

Abstract

In the last years, the number of vehicles on the roads and in the city streets has been increasing. This growing number of vehicles causes congestion on roads and longer travel times, which can multiply increase during the rush hour. Long travel times cause economic losses and make life in big cities uncomfortable. The increasing number of vehicles also causes increased emissions and noise. Therefore, optimization in transportation is gaining importance and significance.

The analytical traffic models are one of the cornerstones of traffic optimization and aim to estimate the future traffic situation based on continuous vehicle flow. The forecasted traffic volumes computed by these models are an important basis for long-term and operational decision-making that tries to optimize the current and future traffic and prevent traffic jams.

This work is focused on analytical traffic models, namely static traffic assignment and dynamic traffic assignment that are widely used in practice for their speed compared to microsimulations. This thesis describes a newly proposed solution method for dynamic user equilibrium, macroscopic model of a node for dynamic network loading, and improves the algorithms for time-dependent shortest path problem and origin-destination matrix estimation.

Abstrakt

V posledních letech se stále zvyšuje počet vozidel na silnicích a v ulicích měst. Narůstající počet vozidel způsobuje přetížení komunikací a prodlužování dojezdových času, které se mohou v dopravní špičce až zněkolikanásobit. Dlouhé dojezdové časy způsobují ekonomické ztráty a znepříjemňují život ve velkých městech. Narůstající počet vozidel také způsobuje zvýšení emisí a hluku. Optimalizace v dopravě nabývá na důležitosti a významu.

Jedním ze základních kamenů v dopravních optimalizacích jsou analytické dopravní modely. Jejich cílem je odhadovat budoucí dopravní situaci na základě spojitého toku vozidel. Budoucí hodnoty objemu dopravy, počítané těmito dopravními modely, jsou důležitým podkladem pro dlouhodobé a operativní rozhodování, jehož úkolem je optimalizovat aktuální a budoucí dopravu a předcházet dopravním zácpám.

Tato práce je zaměřena na analytické dopravní modely, zejména statické přidělování dopravy (Static Traffic Assignment) a dynamické přidělování dopravy (Dynamic Traffic Assignment), které jsou široce využívány v praxi, protože jsou rychlejší než mikrosimulace. Tato práce dále popisuje nově navrženou metodu pro řešení dynamické uživatelské rovnováhy (Dynamic User Equilibrium), makroskopický model pro křižovatky s využitím v dynamickém načítání sítě (Dynamic Network Loading) a vylepšuje algoritmy pro časově závislé hledání nejkratších cest a kalibraci matice přepravních vztahů.

Acknowledgement

I especially would like to thank my supervisor Ivana Kolingerová for leading me in the right direction during my studies. I would also like to acknowledge to Jan Ježek and Taina Haapamäki for introducing me to the topic of transport modeling. My family and girlfriend Šárka Štádlerová also supported me a lot during the study years.

This dissertation thesis was supported by the following projects:

- SGS-2016-004 – Application of Mathematics and Informatics in Geomatics III
  University of West Bohemia (UWB)
- SGS-2019-015 – Application of Mathematics and Informatics in Geomatics IV
  University of West Bohemia (UWB)
- CK01000096 – TRAFFO: Innovative Approaches to Mathematical Traffic Modelling for Sustainable Development of Cities and Regions.
  The Technology Agency of the Czech Republic

Contents

Chapter 1

Introduction

Increasing traffic in cities not only causes increased travel times but also other problems. The congested roads bring higher production of greenhouse gasses and other emissions as well as generate noise. The authorities and road managers responsible for the transport network must solve these issues. The decision on how to improve the situation can be divided into two categories: strategic and operational decisions.

Strategic decisions have long-term consequences and often are connected with an initial investment. The relevant questions of the road managers are: Where should we build a new road and with what capacity so that the effect will be as big as possible? Can we allow some streets to be accessible only to pedestrians without a big impact on congestion in the city? Where can we build new houses or an industrial complex and how does the decision influence the traffic in the city?

Operational decisions try to optimize traffic only using the existing road infrastructure. The relevant issues are: How to coordinate the road reconstruction and closure so that the impact on traffic will be as small as possible? How to set the intersection signal timing plans?

The transportation models and optimization algorithms in transportation help us to answer these questions and to make better decisions. The key properties of the models are precision and computational performance. In general, a higher precision causes a longer computational time. For strategical decisions, we have quite a lot of computational time. On the other hand, operation decisions must be made as quickly as possible, so computational performance is very important and limits the precision. For example, if the forecast for five minutes ahead is computed ten minutes, than the reality comes earlier than the model result. The computational performance is especially important for Intelligent Transportation Systems (ITS).

In this thesis, the algorithms for selected problems in the shortest paths search and suitable transport models are presented. The main goal is to develop more precise models and faster algorithms for selected problems in transportation/traffic modeling.

The analytical transportation models can be divided into two basic categories:

Static Traffic Assignment (or Traffic Assignment Problem) (STA) and Dynamic Traffic Assignment (DTA). This thesis mainly deals with the problems related to the dynamic models where the simulation time plays the key role.

1.1 Problem definition

This thesis aims to improve the performance and precision of the analytical assignments models as dynamic traffic assignment and static traffic assignment.

The first objective is the fast computation of the time-dependent shortest path problem for all departure times with control of precision because the previous algorithms are slow or do not guarantee the maximum error.

The second objective is to design a grid-free and near-time-continuous procedure for the determination of the dynamic user equilibrium. Existing solution methods use time discretization, but nowadays there are a few dynamic network loading procedures that provide near-time-continuous travel times that can be also used with their advantages in equilibrium procedure. Here, the acquired knowledge about the time-dependent shortest path problem is used.

Part of the dynamic network loading procedure is also the node model. The precision of the node model is the third objective. We try to incorporate the capacity of the intersection movements into state-of-the-art models.

The last objective is to develop the method that effectively allows using the bush-based algorithms for origin-destination matrix estimation.

1.2 Summary of contributions

Achieved contributions in this thesis are following:

- *The ε-approximation of the label correcting modification of the Dijkstra's algorithm.* The first contribution focuses on time-dependent shortest path search (TDSP) for all departure times which is an inseparable part of dynamic transportation models. This search consumes a significant part of computational time, therefore, making this search faster brings a significant improvement of DTA model performance. We speeded up the search by approximation of the problem. The continuous-time algorithm that maintains the maximum relative error of the resulting travel times was proposed. Compared to the exact algorithm, the proposed approach saves 97% of memory and 80% of the time in case that the relative error is set to 0.001. The big advantage of our algorithm compared to other effective approximation algorithms is that the error in the resulting travel time is bounded by a maximum value. The algorithm can be also used as a part of precomputation steps of time-dependent shortest path search algorithms as time-dependent contraction

hierarchies. The results were published in (Kolovský et al., 2019b) and (Kolovský et al., 2019a). This contribution is presented in Chapter 2.

- *The piecewise constant/linear solution for dynamic user equilibrium.* The second contribution presents a new solution approach for route-choice dynamic user equilibrium (RC-DUE) where the event-based general link transmission model is used for dynamic network loading (DNL). In the case of analytical DTA models, the general approach is to use the dynamic user equilibrium (DUE) as a route-choice (RC) model. The common approach for solving the RC-DUE problem is to use classic time discretization. These grid-based solutions suffer from classical disadvantages of discretization as non-adaptivity or poor performance in the case of small-time steps. To remove these disadvantages and thus reduce the computational time and increase precision, we designed a solution strategy based on piecewise linear functions instead of time grids. Our proposed strategy is grid-free, near-time-continuous, and computation can be potentially faster because the method computes the results only for times with significant changes. Because of adaptivity, also the memory demand is reduced. This contribution is presented in Chapter 3.

- *Capacity-based macroscopic node model.* The third contribution deals with this capacity-based first-order node model mainly for unsignalized intersections. The dynamic network loading (DNL) provides the propagation of the vehicles through the road network and thus determines the travel times. The important part of DNL is the node model that transfers vehicles through nodes (intersections). The performance and the precision of the node model highly influence the quality of the whole loading procedure. In the case of macroscopic models, the node model has to fulfill the requirements for the class of the first-order node models. Every model belonging to this class takes into account selected factors that influence the number of vehicles transferred by the node. Nevertheless, there are still some situations where the already developed models are not too accurate. The proposed approach reflects the conflicts of the intersections movements as well as the partial first-in-first-out FIFO rule. This contribution is presented in Chapter 4.

- *Origin-destination matrix estimation using bush-based user equilibrium algorithms.* The fourth contribution describes the static origin-destination matrix estimation method that uses the bush-based flow. The key input that is needed for ODM estimation, is the assignment matrix that provides the relationship between edges and OD pairs. This assignment matrix is naturally provided by path-based algorithms. In past years, several very effective bush-based algorithms for the determination of user equilibrium were developed but the assignment matrix is not provided directly by these algorithms. Our proposed approach does not need to enumerate the paths and, therefore saves the memory requirements and

computational time. The savings are most significant in the case of very congested networks. The results were published in (Kolovský and Kolingerová, 2021). This contribution is presented in Chapter 5.

1.3 Thesis structure

Every chapter in this thesis has its introduction, state-of-the-art, and summary section because all selected problems have only a few common parts, terminology, and definitions. The notations are consistent only within one chapter because we try to follow the customs in the individual area of research.

In the whole thesis, the road network is modeled as a directed graph $G = (V, E)$ where V is a set of nodes that represent the intersections and the E is the set of edges that represent the roads and streets.

The approximation of the time-dependent shortest path problem for all departure times and the solution algorithm are presented in Chapter 2. Chapter 3 describes the grid-free and near-time-continuous solution approach for dynamic user equilibrium and shows the advantages of this method. The first-order macroscopic node model for dynamic network loading is proposed in Chapter 4. Chapter 5 presents the method for origin-destination matrix estimation where the bush-based algorithms are used for providing the user equilibrium. Also, the advantages of this implicit method are discussed. Chapter 6 proposes the possible future research directions.

1.4 Application

The algorithm, method, and approaches that are the results of this thesis are incorporated into the software named Traffic Modeler (trafficmodeller.com) which has the aim to provide transportation modeling as a service in your browser. This software is developed together with companies EDIP s.r.o, RoadTwin s.r.o, and HELP SERVICE – REMOTE SENSING s.r.o. in the framework of the H2020 projects OpenTransportNet, PoloVisu (polivisu.eu), DUET (digitalurbantwins.com) and project TRAFFO funded by Technology Agency of the Czech Republic. This solution is currently used by the city of Pilsen for the coordination of road closures and reconstructions.

Chapter 2

Time-dependent shortest path problem

This chapter is focused on searching shortest paths from a source node to all other nodes in a graph with time-dependent edge travel time. This problem is called *time-dependent shortest path problem* (TDSP) and has a lot of applications in transportation. For example, TDSP is one of the sub-problems in dynamic traffic assignment (DTA).

The TDSP can be split into two basic types:

- Compute the arrival time for one given departure time. This can be done using small modification of Dijkstra's algorithm.
- Compute the arrival time for a departure time interval (profile search).

In the upcoming text, we deal with the second type problem. More formally, given a directed graph $G = (V, E)$ and a source node $s \in V$, we want to know the travel time between the source node s and all other nodes for every departure time (in some literature the task is called a *travel time profile*). The main principle is that the arrival time t_u at the node u is used as the argument of the arrival time function f corresponding to the edge that origins at u. The common approach is to use a piecewise linear function as a realization of the arrival function.

As mentioned-described in Chapter 1, the main problem is that the searching takes a lot of time and needs a lot of memory because the number of linear pieces of arrivals functions highly increases with the length of paths. These properties were the reason for moving to the approximation algorithms. There are two approaches how to approximate the solution. The first group of algorithms directly simplifies the result arrival functions that were computed by the exact label correcting algorithm. These approaches guarantee only a maximal error of further computations dependent on the level of approximation and do not fully remove the problem with the increasing number of linear pieces. The second group of methods computes directly the approximation

of the exact solution with a guarantee of maximal allowed error but these methods perform more than one static shortest path search per linear piece of all arrival functions associated with edges. Our algorithms that are based on the label correcting algorithm simplify the arrival function during the searching and thus prevent rising the number of linear pieces. Using the appropriate dynamic choice of simplification level, the algorithm also provides the solution with the maximally allowed error that was defined before computation.

Section 2.1 formally defines the problem and presents the state-of-the-art exact methods for solving it. The ε-approximation of the problem is defined in Section 2.2. Section 2.3 shows the comparison of exact algorithms with proposed algorithms on real road networks.

2.1 Definitions and preliminaries

Let $G = (V, E)$ be a directed graph that represents a road network, where V is a set of nodes and E is a set of edges. Each edge $(u, v) \in E$ has an *arrival function* (AF) $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ that for the given departure time at u returns the arrival time at v. Alternatively, we can define a *travel time function* (TTF) that returns the time needed to cross the edge. A relationship between the TTF g and the corresponding AF f is defined as $g(t) = f(t) - t$.

It is assumed that every AF f fulfill the FIFO property: $\forall t_1 < t_2 : f(t_1) \leq f(t_2)$ and the departure time t_d must be smaller then the arrival time t_a (the travel time must be positive). AFs are implemented as piecewise linear functions.

In Figure 2.1a you can see AFs (f_{su}, f_{sv}, f_{ud} and f_{vd}) for every edge in a small example graph with four nodes $V = \{s, u, v, d\}$ and four edges $E = \{(s, u), (s, v), (u, d), (v, d)\}$.

The points of AFs are called *breakpoints*. The number of breakpoints of AF f can be written as $|f|$. The following operation must be defined for two AFs:

- There are two consecutive edges (s, u) and (u, d) with AFs f_{su}, f_{ud}. The operation *combination* $f_{ud} * f_{su} : t \mapsto f_{ud}(f_{su}(t))$ represents AF from s to d. In Figure 2.1b there are AFs as results of the combination along the paths (s, u, d) (solid red line) and (s, v, d) (solid blue line).

- There are two parallel paths p_1, p_2 from s to d with AFs f^1_{sd}, f^2_{sd}. The operation *minimum* $\min(f^1_{sd}, f^2_{sd}) : t \mapsto \min\{f^1_{sd}, f^2_{sd}\}$ represents the earliest AF from s to d. In Figure 2.1a $p_1 = (s, u, d)$ and $p_2 = (s, v, d)$. In Figure 2.1c you can see this earliest AF as a result of the operation minimum (green line).

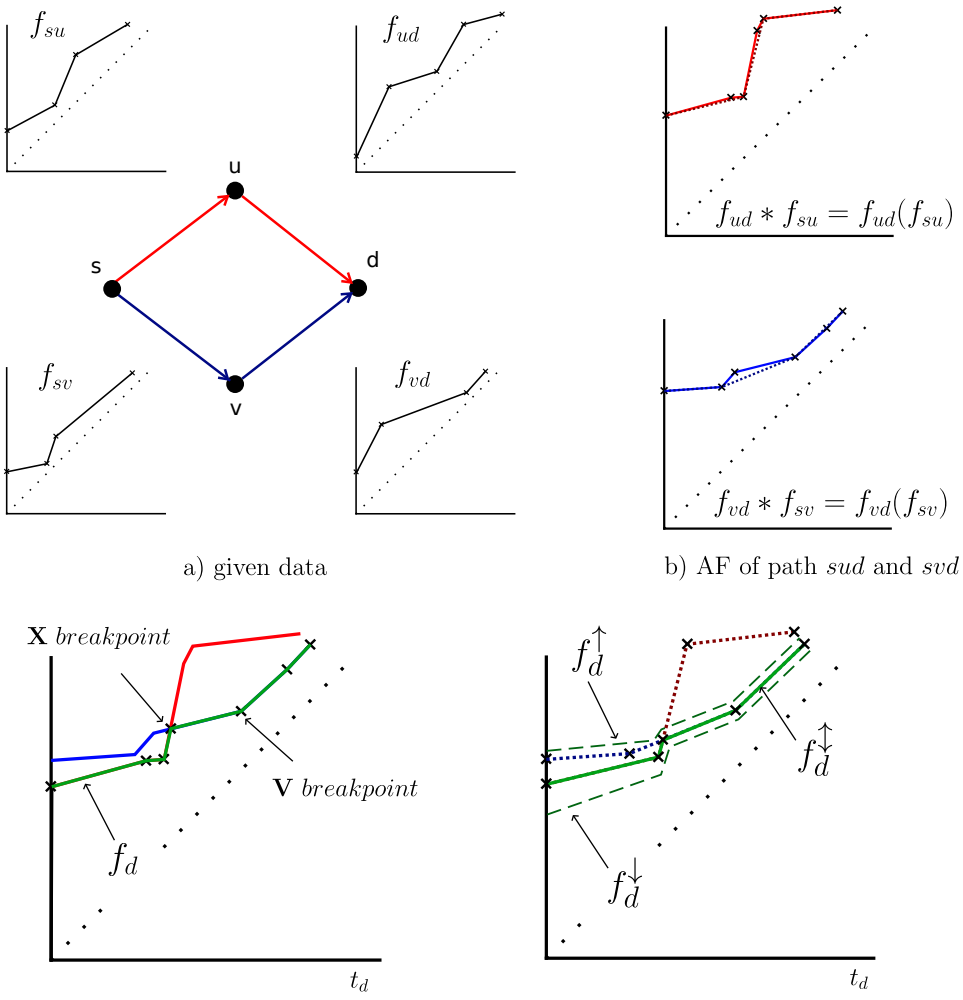


Figure 2.1: Example of calculation of the arrival function from node s to d

2.1.1 Problem definition

More precisely, TDSP can be defined as minimizing the travel time over the set $P_{s,d}$ of all paths in G from the source node s to the destination node d:

$$f_d = \min\{f_p(t) | p \in P_{s,d}\} \tag{2.1}$$

where f_d is the function of the earliest arrival time (minimal AF) from s to d and f_p is AF of the path $p \in P_{s,d}$.

This chapter deals with *one-to-all* problem. The input data are the graph G, AF f_{uv} for every edge $(u, v) \in G$ and the source node s. The output is the set F of the earliest AFs from the source node s to all other nodes u: $F = \{f_u | u \in V \setminus \{s\}\}$.

2.1.2 Related work with exact problem

In 1990, the Orda and Rom (1990) published the label correcting algorithm (LCA), see Algorithm 1, that is the modification of the Dijkstra's algorithm for time-dependent networks. The modifications are:

- The node labels are AFs from s.
- The key of the priority queue is the minimum of AF ($\min f$).
- The relaxation of the edge (u, v) is performed using $f_v = \min(f_v, f_{uv} * f_u)$.

Algorithm 1: LCA in the exact form

```
1  PQ = minimum priority queue where key is min f
2  ∀u ∈ V : f_u = ∞
3  g_s = 0
4  PQ.put(f_s)
5  while queue is not empty do
6      f_u = PQ.get()
7      foreach v : (u, v) ∈ E do
8          f̄_v = f_uv * f_u                          // combination
9          if ∃t : f̄_v(t) < f_v(t) then               // compare
10             f_v = min(f̄_v, f_v)                    // min
11             PQ.put(f_v)
```

In Algorithm 1, the initialization is performed in the lines 1-4. All node labels (AFs) are set to infinity in the line 2. The travel time at s is set to zero (line 3) and the node

label at s (f_s) is added to the priority queue (PQ) (line 4). In the line 6 the algorithm takes the node on the top of PQ and relaxes all edges that lead from this node. The relaxation is represented by the lines 8-11. The line 8 performs the combination of the node label at u (f_u) and the edge AF (f_{uv}). The condition in the line 9 checks for update. Updates of the label at the node v are performed in the line 10. The line 11 puts the node v to the PQ. The time complexity of the LCA is $O(|V||E|P_{max})$, where P_{max} is the maximum number of linear pieces needed to represent the earliest arrival time function.

Dean (1999, 2004) presents the Label setting algorithm that improves the worst-case running time to $O(|E|P^{**}log|V|)$, where P^{**} is the total number of pieces of the results functions. The running time is highly dependent on the number of pieces in the result. This author also discusses the possibility of parallelization.

Very similar approach is used by Ding et al. (2008), the time complexity of their algorithm is $O((|V|log|V| + |E|)\alpha(T))$, where $\alpha(T)$ is is the cost required for each function operation.

Dehne et al. (2012) use a different approach. The asymptotic time complexity of their algorithm is $O((F_d + \gamma)(|E| + |V|log|V|))$, where F_d is the number of the linear pieces of the result functions and γ is the total number of the linear pieces needed to represent the input arrival functions. This algorithm use *forward* and *backward probes*. The forward probe computes the arrival time at the node d with the given departure time at the node s. The backward probe solves the inverse problem. The arrival time at d is given and we want to know the departure time at s. These probes can be computed using the well-known Dijkstra's algorithm.

This method recognizes two types of breakpoints. The **V** points represent points that are created as images of the breakpoints that lie on the edge arrival functions $\{f_{uv}|(u, v) \in E\}$. The **X** points are created as an intersection of two AF in the *minimum* operation. The algorithm consists of two steps:

- First, the **V** breakpoints are computed. For every breakpoint in the edge arrival functions, two forward probes and one backward probe are performed.
- After that, the **X** breakpoints are computed using intersections. This step can take a non-polynomial time (Foschini et al., 2014).

Foschini et al. (2014) proved that TDSP in the exact form is non-polynomial.

2.2 Approximation of the problem

Let us have two consecutive edges (e.g, the edges (s, u) and (u, d) in Figure 2.1a) then the arrival time at the node d is the value of the arrival function f_{ud} in the arrival time $f_{su}(t_d)$ at the node u, where t_d is the departure time at the node s. In the exact case,

the combination $f_2(f_1(t))$ of two piecewise linear functions f_1, f_2 with $|f_1|$, $|f_2|$ linear pieces is also a piecewise linear function with up to $|f_1| + |f_2|$ linear pieces (Foschini et al., 2014). It means that the arrival function at the end of the path with n edges can have up to $\sum_{i=1}^{n} |f_i|$ linear pieces. For example, the path across the Pilsen city has around 100 edges. If every arrival function on the path has 24 linear pieces, the resulting arrival function has 2400 linear pieces. It can be seen that the computational time and memory requirements strongly increases with the length of the paths.

2.2.1 Related work with approximation

There are two groups of methods that compute an approximation of the AF. The first group uses the idea of *forward* and *backward probes* by Dehne et al. (2012). It can be proved that the AF between two consecutive **V** points is concave or a line segment (Foschini et al., 2014) (see Example in Fig.2.1c). The algorithms described by Foschini et al. (2014) and Omran and Sack (2014) use this concavity. First the **V** points are computed using one backward probe and two forward probes and then the approximation of AF between the **V** points is determined. The main problem of this approach is that the computation of **V** points requires $3\sum_{(u,v)\in E} |f_{uv}|$ probes (Dehne et al., 2012).

The second group of methods uses a label correcting modification of the Dijkstra's algorithm (LCA) (Algorithm 1). LCA has time complexity $O(|V||E|P_{max})$ (Orda and Rom, 1990), but a real road network is far from the worst case. This technique is widely used, see e.g, (Geisberger and Sanders, 2010; Batz et al., 2013; Geisberger, 2010). First LCA is performed in the exact form and after that the resulting arrival functions F are simplified and used for further computation (e.g, some query algorithm). Some guarantees about the error of AF are presented by Geisberger and Sanders (2010), but these guarantees give only a maximum error dependent on the degree of approximation.

The main task is to develop an algorithm which solves TDSP with the given maximum relative error and is effective for a real road network. It follows that we focused on the ε-approximation of the LCA.

2.2.2 The proposed approximation

The proposed algorithm is based on the so-called label correcting modification of Dijkstra's algorithm by Orda and Rom (1990). The main idea is to perform simplification of the arrival functions during the computation with a suitable maximum absolute error such that the relative error ε is maintained.

The ε-approximation of AF $f^{\updownarrow}$ is understood as the ε-approximation of TTF $f(t) - \varepsilon g(t) \leq f^{\updownarrow}(t) \leq f(t) + \varepsilon g(t)$. So the ε-approximation of the set F is $F^{\updownarrow} = \{f_u^{\updownarrow} | u \in V \setminus \{s\}\}$.

Let us present some useful theorems about the approximation that were derived for a use in the proposed algorithm.

Theorem 1. *Let $g^{\updownarrow}$ be an ε-approximation of TTF g. Then it holds that*

$$g^{\downarrow} = \frac{g^{\updownarrow}}{1+\varepsilon} \leq g \leq \frac{g^{\updownarrow}}{1-\varepsilon} = g^{\uparrow}$$

Proof. If the function $g^{\updownarrow}$ is substituted by its extreme values $(1-\varepsilon)g$, $(1+\varepsilon)g$, the expression is still valid. □

Theorem 2. *Let $f_u^{\updownarrow}$ be an ε-approximation of AF f_u and $\overline{f}_v^{\updownarrow} = f_{uv} * f_u^{\updownarrow}$. Let $\alpha \in [0, \frac{\overline{g}_v^{\downarrow}}{g_u^{\downarrow}}]$ be the maximum slope of AF f_{uv}, $approx(f, \delta)$ be a function that simplifies the AF f with the maximum absolute error $\delta \geq 0$. Then $f_v^{\updownarrow} = approx(\overline{f}_v^{\updownarrow}, \varepsilon\overline{g}_v^{\downarrow} - \alpha\varepsilon g_u^{\downarrow})$ is the ε-approximation of AF f_v.*

Proof. The maximum absolute error of f_v is εg_v and the maximum absolute error of the operation $f_{uv} * f_u^{\updownarrow}$ is $\alpha\varepsilon g_u$. Then the result of the combination can be simplified with the maximum absolute error:

$$\delta = \varepsilon g_v - \alpha\varepsilon g_u \geq \varepsilon\overline{g}_v^{\downarrow} - \alpha\varepsilon g_u^{\downarrow}$$

Then δ must be ≥ 0

$$\varepsilon\overline{g}_v^{\downarrow} - \alpha\varepsilon g_u^{\downarrow} \geq 0$$
$$\alpha \geq \frac{\overline{g}_v^{\downarrow}}{g_u^{\downarrow}}$$

□

Theorem 2 can be also formulated in a local form for a given departure time. In Figure 2.1b there are dotted lines that represent the approximation of AFs. In Figure 2.1d you can see the ε-approximation $f_d^{\updownarrow}$ of the earliest AF f_d from s to d (solid green line) and its upper bound $f_d^{\uparrow}$ and lower bound $f_d^{\downarrow}$ (green dashed lines).

2.2.3 e-LCA algorithm

The basic idea of the first proposed algorithm (ε-LCA) is that the simplification of AF is performed after every edge relaxation (the operation combination). The degree of the simplification is directed by Theorem 2.

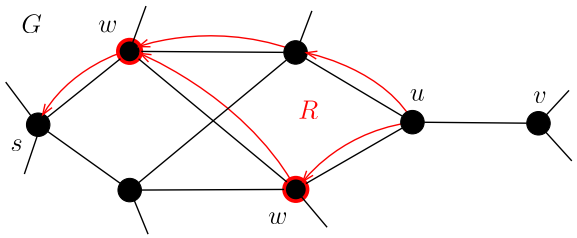


Figure 2.2: Example of backsearch procedure

The ε-LCA computes AF with the maximum relative error ε assuming that

$$\forall (u,v) \in E : \alpha_{uv} \in \left[0, \frac{\overline{g}_v^{\downarrow}}{g_u^{\downarrow}}\right] \tag{2.2}$$

where α_{uv} is the maximum slope of AF f_{uv}. The slope α_{uv} must be bounded because Theorem 2 is used in the ε-LCA and the theorem needs this assumption.

The ε-LCA differs from the exact LCA only in the computation of $\overline{f_v}$. The AF $\overline{f_v}$ in the line 8 in Algorithm 1 is simplified with the maximum absolute error δ (according to Theorem 2). So the line 8 is replaced by 2 lines

$$\begin{aligned} \delta(t) &= \varepsilon((f_{uv} * f_u^{\updownarrow})^{\downarrow}(t) - t) - \alpha(t)\varepsilon g_u^{\downarrow}(t) \\ \overline{f_v} &= approx((f_{uv} * f_u^{\updownarrow}), \delta) \end{aligned} \tag{2.3}$$

where $\alpha(t)$ is the maximum slope of f_{uv} in the interval $[f_u^{\downarrow}(t), f_u^{\uparrow}(t)]$. The simplification was performed using Douglas-Peucker algorithm or Imai and Iri algorithm (Imai and Iri, 1986).

The main problem of ε-LCA is that if the assumption (2.2) is not complied, $\delta < 0$ and the algorithm cannot ensure the given relative error ε. This occurs when the maximum slope α_{uv} is too large. This issue is resolved using the second algorithm that is described in the upcoming section.

2.2.4 e-LCA-BS algorithm

The second proposed algorithm (ε-LCA-BS) is based on backsearch. It has no limitations for the slope α. The pseudo-code of the ε-LCA-BS is in Algorithm 2. The basic idea is that if the algorithm finds an edge (u,v) where α is too big in some departure time interval $[t_m, t_n]$ ($\delta < 0$), it determines f_v in $[t_m, t_n]$ again with a higher accuracy (lines 11-15 of the Algorithm 2). The algorithm returns back to the point such that the edge (u,v) can be reached from this point with sufficient precision using the exact LCA.

When the label at the node v (AF f_v) is updated (the condition at the line 16 is fulfilled), the edge (u,v) is added to the predecessor list $pred(v)$ of the node v (lines 17-20). The set of predecessors form a graph $R = (V_R, E_R)$ (red color in Fig. 2.2). We

Algorithm 2: ε-LCA-BS

PQ = *minimum priority queue where key is* $\min f$
$\forall u \in V : f_u^{\updownarrow} = \infty$
$g_s^{\updownarrow} = 0$
PQ.put($f_s^{\updownarrow}$)
pred(s) = null
while *queue is not empty* **do**
  $f_u^{\updownarrow}$ = PQ.get()
  foreach $v : (u, v) \in E$ **do**
    $\delta(t) = \varepsilon((f_{uv} * f_u^{\updownarrow})^{\downarrow}(t) - t) - \alpha(t)\varepsilon g_u^{\downarrow}(t)$ `// according equation 2.3`
    $\overline{f}_v = approx(f_{uv} * f_u^{\updownarrow}, \delta^+)$ `//` $\delta^+(t) = \max(\delta(t), 0)$
    if $\exists t : \delta(t) < 0$ **then**
      find all intervals $[t_m, t_n]$ *where* δ *is negative*
      foreach $[t_m, t_n]$ **do**
        $h = backSearch((u, v), [t_m, t_n])$
        substitute $\overline{f}_v$ *by* h *in interval* $[t_m, t_n]$
    if $\exists t : \overline{f}_v(t) < f_v^{\updownarrow}(t)$ **then**
      if $\overline{f}_v < f_v^{\updownarrow}$ **then**
        pred(v) = (u, v)
      else
        pred(v).add((u, v))
      $f_v^{\updownarrow} = \min(\overline{f}_v, f_v^{\updownarrow})$
      PQ.put($f_v^{\updownarrow}$)

assume that the graph R is acyclic. In general, the graph R may not be acyclic, but in real case it is very unlikely.

We want to find nodes w such that if the exact LCA is performed from these nodes w, the AF f_v is an ε-approximation. The nodes w have to satisfy the following inequalities in the interval $[t_m, t_n]$:

$$\max\left\{\prod_{e\in p}\alpha_e | p \in P_{vw}\right\} \leq \min\left(\frac{\overline{g}_v^{\downarrow}}{g_w^{\downarrow}}\right) \tag{2.4}$$

where P_{vw} is the set of all paths from v to w in the graph R (the paths must contain the edge (u, v)) and α_e represents the maximum slope of AF f_e corresponding with the edge e. The set W is the set of all nodes w that meet the condition (2.4) and there is a

path $p \in P_{vw}$ that does not contain any other node from the set W (W is the smallest possible).

These nodes w can be found using a topological ordering of V_R (Algorithm 3). The node labels α_b correspond to the left side of the inequalities (2.4). So the algorithm finds maximal paths in R. First the labels are set to negative infinity (the line 2) and the label at u is set to α_{uv} (the line 3). The lines 4-5 ensure a topological ordering. If the condition (2.4) in the line 6 is fulfilled, b is added to the W. The lines 9-12 ensure updating of the node labels. The part of $\overline{f}_v$ in the interval $[t_m, t_n]$ is substituted by a more accurate result of the exact LCA with the initial priority queue PQ that is created by adding all $f_w \in \{f_w | w \in W\}$ (the lines 13-16).

In Figure 2.2 there is an example of backsearch. The black color represents the original graph G and the red color represents the acyclic graph R. The edge (u, v) violates the condition 2.2. Then the algorithm starts *backSearch* procedure and finds the set W (red nodes) using the graph R.

Algorithm 3: backSearch

```
1  W = {}                                                    // set of all w
2  ∀b ∈ V_R : α_b = −∞
3  α_u = α_uv
4  while ∃b : b ∈ V_R \ W ∧ deg⁻(b) = 0 do                   // topological ordering
5      b = some node that meets the conditions above
6      if α_b ≤ min( g_v↓ / g_b↓ ) then
7          W = W ∪ {b}
8      else
9          foreach a : (b, a) ∈ R do
10             if α_b α_ba > α_a then
11                 α_a = α_b α_ba
12         V_R = V_R \ {b}
13 foreach w ∈ W do
14     PQ.put(f_w)
15 run LCA on G with initial priority queue PQ on interval [t_m, t_n] // algorithm
       1
16 return f_v
```

When the graph R is not acyclic, it is necessary to modify the algorithm for searching the set W.

If the condition (2.2) is fulfilled, the ε-LCA-BS is reduced to ε-LCA, because the algorithm then does not perform any *backSearch* procedure.

2.2.5 Heuristic improvements

The proposed solution can be further accelerated on the basis of departure time interval decomposition as follows. The profile search problem on the departure time interval $[a, b]$ can be decomposed in time to two subproblems on intervals $[a, c]$ and $[c, b]$ and these subproblems are independent (Dean, 2004). It follows, the problem can be split into N independent computation parts. This property enables effective parallelization and distribution of the computation.

A surprising property is that the splitting of the problem often saves total computation time in the serial case, too. The ε-LCA-BS runtime on the interval $[a, b]$ is often higher than the sum of runtimes on the intervals $[a, c]$ and $[c, b]$. In the following text, the reasons why the decomposition is faster than the original problem are presented. Some splitting strategies are also discussed.

The number of relaxations K (the number of iterations of the main loop) of ε-LCA-BS is greater than or equal to the number of relaxations of the Dijkstra's algorithm (Orda and Rom, 1990). The aim is to approach as close to the Dijkstra's algorithm as possible. The condition at line 16 in Algorithm 2 controls adding to the priority queue (PQ) and thereby controls the number of iterations. The condition compares two AFs. The small piece of the $\overline{f}_v$ under $f_v^{\updownarrow}$ is enough for adding to the PQ. It follows, a more fluctuating TTFs indicate a higher K.

The fluctuation of the TTFs can be reduced by splitting the function to intervals. The function on a shorter interval has smaller or equal difference between its maximum and minimum. It means that in more cases the condition $\exists t : \overline{f}_v(t) < f_v^{\updownarrow}(t)$ is false. In Figure 2.3 there is an example of splitting. The original function $\overline{f}_v(t)$ on the interval $[a, d]$ is divided into 3 intervals $[a, b]$, $[b, c]$ and $[c, d]$. As you can see, for the intervals $[a, b]$ and $[c, d]$ the condition at line 16 in Algorithm 2 is false. Thus, the node will not be added to the PQ and K is reduced.

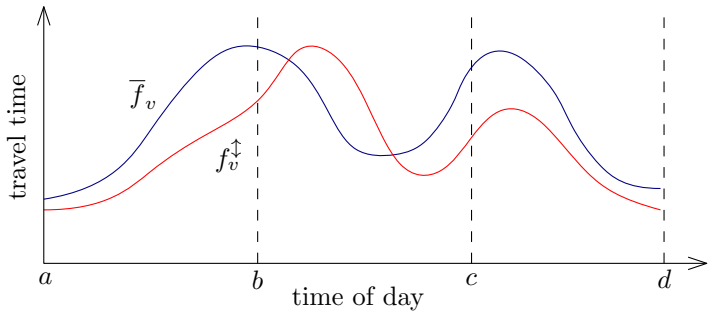


Figure 2.3: Splitting effect

The question is how to split the departure time interval so that the computation is the most effective. In Figure 2.4 there is a visualization of speed profiles for the testing dataset. The x-axis represents a time of day and the y-axis displays the average speed

in time on edges. In the visualization, all speed profiles from the tested dataset are rendered in various colors. As you can see, the fluctuation in the night time (the first part and the last part) is small. There are two main concepts for splitting:

- Split the origin interval into equal subintervals.
- Split the origin interval into inhomogeneous subintervals (e.g., longer at night and shorter by day).

Which of these concepts is better strongly depends on the TTFs shape, as further presented in experiments.

The splitting into intervals enables a very simple parallelization. Due to the independence in the departure time intervals, we can set one interval equal to one thread.

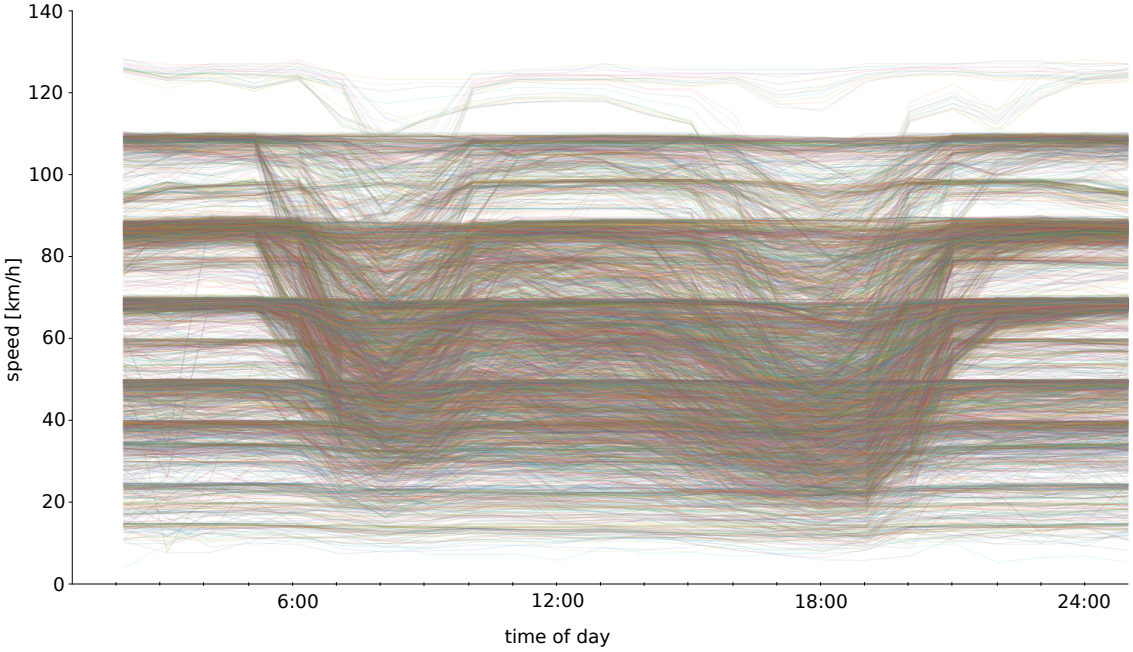


Figure 2.4: Speed profiles vizualization for datset G1

2.3 Experiments

The real road network with real speed profiles that were computed from GPS tracks was used for testing. These data represent part of Paris (Figure 2.5).

The algorithms were implemented using Scala programming language (OpenJDK 1.9, Debian 11). The testing was performed on a computer with Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz and with 16 GB RAM.

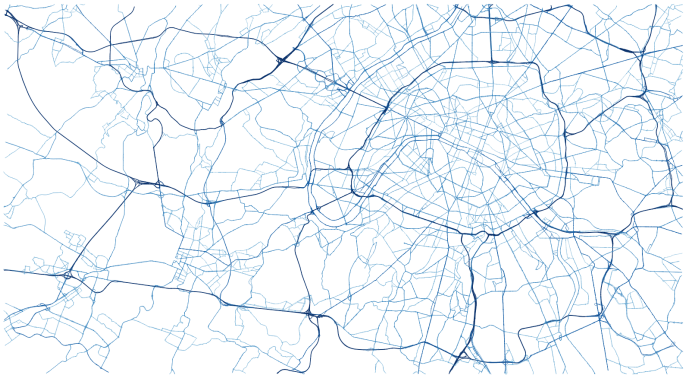


Figure 2.5: Route network for testing

2.3.1 The ε-LCA-BS testing

Only the ε-LCA-BS was tested because ε-LCA is only a special case of ε-LCA-BS. The maximum allowed relative error ε was set to $10^{-2}, 10^{-3}, 10^{-4}$ and 10^{-5}.

Four graphs (G1, G2, G3 and G4) were created to show the performance of the developed algorithms. Every edge in the graphs has AF with 24 linear pieces. Every graph represents different classes of roads. In Table 2.1 there are numbers of edges for each graph and performance results of ϵ-LCA-BS: the relative time t_r and the relative number of breakpoints *bps* related to the exact version of LCA. The same results you can see in Figure 2.6.

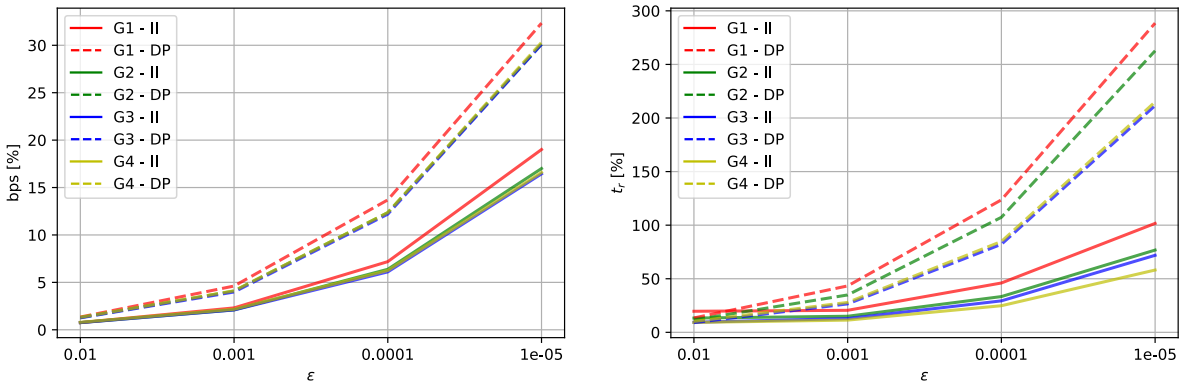


Figure 2.6: The relative number of breakpoints and the relative time related to exact LCA (II - Imai and Iri, DP - Douglas Peucker)

The results in Table 2.1 show that the maximum relative error 10^{-4} brings only a small improvement and the maximum relative error 10^{-5} is slower than the exact version of the LCA, but this accuracy is too big for a real use. The maximum relative error from 10^{-2} to 10^{-3} seems to be a good compromise between accuracy and performance.

dataset	#edges	ε	Imai and Iri		Douglas Peucker	
			t_r [%]	*bps* [%]	t_r [%]	*bps* [%]
G 1	10 798	10^{-2}	19.7	0.8	13.4	1.3
		10^{-3}	20.6	2.3	43.3	4.6
		10^{-4}	46.0	7.2	123.6	13.7
		10^{-5}	101.7	19.0	288.5	32.3
G2	33 354	10^{-2}	13.3	0.8	11.6	1.3
		10^{-3}	15.0	2.1	34.9	4.1
		10^{-4}	33.4	6.4	107.3	12.4
		10^{-5}	76.8	17.0	262.7	30.1
G3	107 476	10^{-2}	9.4	0.8	9.0	1.3
		10^{-3}	12.8	2.1	26.5	4.0
		10^{-4}	29.4	6.1	82.1	12.2
		10^{-5}	71.9	16.4	211.6	30.1
G4	160 092	10^{-2}	9.2	0.8	10.2	1.3
		10^{-3}	11.4	2.2	27.9	4.1
		10^{-4}	24.8	6.3	84.8	12.4
		10^{-5}	58.1	16.6	215.0	30.3

Table 2.1: Results of testing ϵ-LCA-BS (t_r - the relative time related to the exact version of LCA, *bps* - the relative number of breakpoints, ε - the maximum allowed relative error)

	Imai and Iri		Douglas Peucker	
	time [s]	# bps	time [s]	# bps
G1	0.9	561 853	1.8	1 122 458
G2	3.0	1 429 080	5.6	2 779 762
G3	9.3	3 855 536	18.8	7 352 745
G4	14.1	5 298 280	28.8	10 057 055

Table 2.2: Absolute values of measured parameters for $\varepsilon = 0.001$

In Table 2.2 there are absolute values of measured parameters for the maximal relative error 10^{-3}. The column *bps* represents the number of breakpoints in the resulting AFs.

The results show that the breakpoints savings are significant. It means that the ε-LCA-BS saves a lot of memory. Let us assume a path that takes 1 hour, then the relative error 0.1 % implicates the absolute error 3.6 s. In this case the epsilon approximation saves more than 97% of memory and 80% of time. In case that ε is too small then the algorithm can run slower than the exact version, because the simplification takes too much time.

The main disadvantage of the ε-LCA-BS is that it is sensitive to values of the maximum slope of AFs. If AFs have too big slope then the algorithm performs too many calls of *backSearch* procedure and thereby makes the computation too slow. In practice, the functions usually have small slopes. When it is certain that input data do not violate the condition (2.2), the algorithm is more suitable.

Furthermore, the computation of $\mathbf{V}$ breakpoints was implemented. This computation is the first step of the all algorithms presented by Foschini et al. (2014), Dehne et al. (2012), Dehne et al. (2009), and by Omran and Sack (2014). As mentioned above, the step needs $3\sum_{(u,v)\in E}|f_{uv}|$ static shortest path computation. The $\mathbf{V}$ breakpoints determination was performed on G1 and takes 21 minutes.

2.3.2 Splitting tests

The first test is focused on equal subintervals. The original departure time interval was divided into 2 to 12 subintervals. The two datasets (G1 and G2) were used and the maximum relative error was set to 10^{-2}, 10^{-3}, and 10^{-4}. The more effective simplification method by Imai and Iri was set for all tests in this section. The test was performed only in serial (one thread was used only). In Figure 2.7 there are results of the test. The chart shows the dependence between the speed-up and the number of subintervals. The speed-up is defined as:

$$\text{speed-up}(N) = \frac{\text{runtime for 1 interval}}{\text{runtime for N subintervals}} \tag{2.5}$$

As you can see, the speed-up is between 1.1 and 1.8 and is very variable, but never under 1. For our testing data, 4 -10 equal subintervals are a good choice. In case when the number of subintervals is too big (in our case more than 10), the overhead costs override the benefits of splitting.

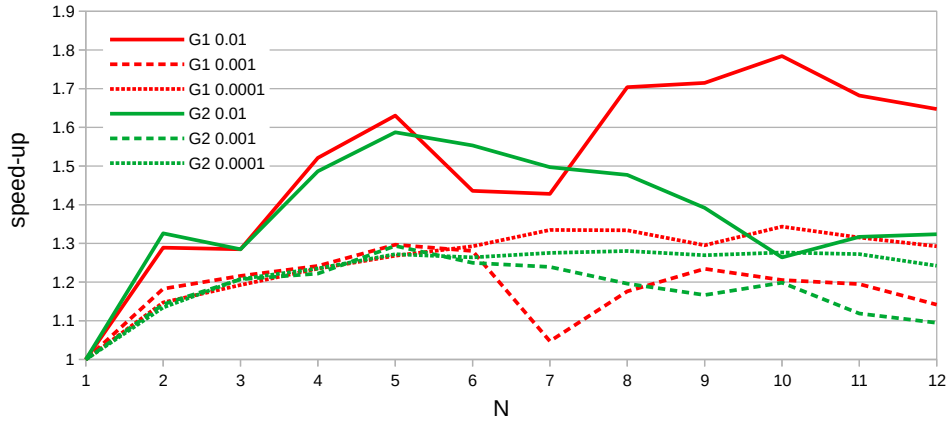


Figure 2.7: Speed-up dependence on the number of subintervals N in serial case (1 thread)

The inhomogeneous subintervals were also tested. Several methods for the departure time interval splitting were tried, but the runtime was only a few percents better than the splitting into equal subintervals. The conclusion is that the equal subintervals, although also more simple, work better.

The last test demonstrates the suitability of parallelization. The dataset G1 with $\varepsilon = 0.001$ was used. The algorithm was performed with 3 settings: without splitting into subintervals, with 4 subintervals in the serial case (1 thread) and with 4 subintervals in 4 threads. The code was performed on one computer with multi-core processor. The following runtimes were measured (Table 2.3):

1 interval, 1 thread	4 intervals, 1 thread	4 intervals, 4 threads
8.6 s	7.6 s	2.6 s

Table 2.3: Paralellization runtimes on G1 with $\varepsilon = 0.001$

The testing shows that the parallelization is suitable so it is recommended.

2.3.3 Summary

The algorithms significantly reduce the memory use. When the maximum relative error is a sufficiently large value (in our case 10^{-3}), the algorithms save the computational time, too. From this point of view, the algorithms are suitable for precomputing the

TTFs for the next use (e.g., time-dependent distance oracles, time-dependent contraction hierarchies). In a real road network the maximum slopes of AFs are not too big Strasser (2017). So the main disadvantage (too many calls of back search procedure) is not a too big problem. It has been shown that the splitting into departure time subintervals can further reduce the runtime and this splitting is suitable for parallelization or distribution because the subproblems are independent.

Chapter 3

The piecewise constant/linear solution for dynamic user equilibrium

This chapter presents the method that tries to increase the precision of the solution of dynamic user equilibrium (DUE) assuming that the computation takes a reasonable time so that the solution is useful in practice. The proposed method replaces the classical grid-based solution by a near continuous-time solution based on piecewise linear/constant functions that removes a lot of disadvantage of discretization. The important part of DUE computation is searching of time-dependent shortest paths. For this purpose, we use the continuous algorithm that was described in Chapter 2. The presented method was sent for publication in Networks and Spatial Economics journal.

3.1 Introduction

The predicted traffic situation is the output of the models. It includes information such as the traffic volume and the travel time. The goal is to make the model precise enough so that it takes into account the behavior of a real route network and drivers. To be used in the Intelligent Transportation Systems (ITS), the result should be determined within a reasonable time. The approach that tries to meet these requirements is called Dynamic Traffic Assignment (DTA).

The DTA model consists of two submodels. The dynamic network loading (DNL) maintains the flow propagation through the road network. The second submodel deals with the user route choice. The classical approach is to load the network using DNL and after that to update the information about the route choice using the travel time. These two procedures are repeated until the precision of the result is sufficient.

This whole scheme uses the time discretization, so it is a grid-based solution. The

common problem of the grid-based solutions is that in some cases the algorithm repeats the same computation and stores redundant data. If there is a long time interval where nothing happens, the grid solution computes and stores the same values again and again. On the other hand, small changes are ignored. It follows that the optimal time step for discretization is hard to estimate.

Another problem is the discretization itself. Nowadays the common approach for DNL is to use the first-order macroscopic traffic model that is extended to a network. The most used DNL model is Link Transmission model (LTM) and its derivatives. In the case of LTM, the size of the time step is very restricted, e.g., the shortest edge in the graph must be longer than the time step.

In the last years several papers that provide a grid-free continuous-time solution for the DNL within a reasonable time were published. So the DNL submodel is ready for a continuous-time DTA. The problem is that the route choice procedure is still grid-based and thus it does not take advantage of the continuous-time solution of grid-free DNL. In this chapter the method that tries to solve this issue is proposed.

This chapter defines and solves the continuous-time route-choice (RC) dynamic user equilibrium (DUE) based on piecewise linear and constant functions. It follows that all the parts of the solution are continuous-time and also grid-free. These parts are a time-dependent shortest path search (profile search) and an update of the route choice information. The exact continuous-time solution based on piecewise function is very demanding so the approximation and simplification of the solution are also discussed here.

Section 3.2 presents the relevant state-of-the-art about DNL and DUE. The main concept of the proposed solution and basic definitions are in Section 3.3. Section 3.4 presents the developed adjustments and extensions for event-based dynamic network loading algorithm that are necessary for intersection-movement-based formulation of the problem. The piecewise linear solution for time-dependent shortest path problem is discussed in Section 3.5.

3.2 State of The Art

The following text contains a preview of the previous work and discusses existing solutions for DNL and DUE models. The DUE in this chapter is understood as a route-choice (RC) DUE because this model is the simplest and is widely used in practice.

3.2.1 Dynamic Network Loading

The Dynamic Network Loading is a procedure that ensures the propagation of vehicles through the network. The input is a time-dependent origin-destination matrix (ODM) and output is a travel time function for every edge. The DNL is sometimes called a

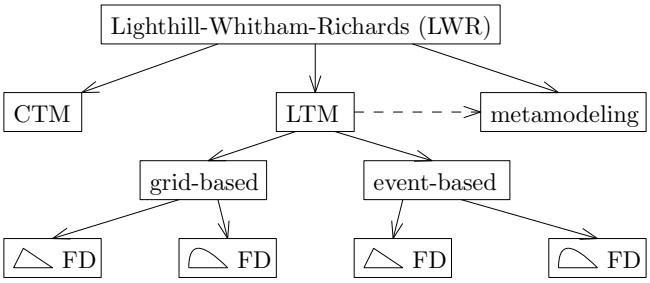


Figure 3.1: Categorization of the DNL methods

delay operator. The DNL should respect the properties of the real road network as queues and spillback. According to Hoogendoorn and Bovy (2001), the flow models can be subdivided into three categories: microscopic, mesoscopic and macroscopic. For DNL purposes these flow models must be extended to the network.

The macroscopic Lighthill-Whitham-Richards (LWR)-based models (Lighthill and Whitham, 1955), (Richards, 1956) are used in most analytical DTA solutions. In Figure 3.1 there is a categorization of these methods. The models based on LWR kinematic wave theory are: Cell Transmission Model (CTM) by Daganzo (1994, 1995) and the Link Transmission Model (LTM) family.

The LTM family can be divided into four groups according to the shape of Fundamental Diagram (FD) and to the solution method.

- *The grid-based solution with a triangular FD* is the classical approach represented by Yperman's LTM (Yperman, 2007) and its modification and extensions. Nowadays, this approach is used in a lot of solutions, e.g., (Nezamuddin and Boyles, 2015), (Han et al., 2018). The LTM is also a special case of a solution of differential algebraic equations by Han et al. (2016). Main disadvantage is that the size of the time step is very restricted. This issue was solved by the Iterative Link Transmission Model (ILTM) by Himpe et al. (2016) that allows longer time steps, thereby reducing time and memory requirements, but other disadvantages of discretization persists.

- *The grid-based solution with a non-triangular FD* moves the model closer to reality. This model is called General LTM (GLTM). The linear parts of the triangular FD are replaced by continuous concave parts. A dual quadratic FD is used most often. The GLTM is used by Gentile (2015, 2016) and van der Gun et al. (2017). Himpe (2016) also presented a solution with non-linear FD for one link. The disadvantages are the same as in the previous category.

- *The event-based solution with a triangular FD* is a completely different approach. The main idea of this approach is to track every flow change how it is propagated through the network. Every flow change in the network is called an event.

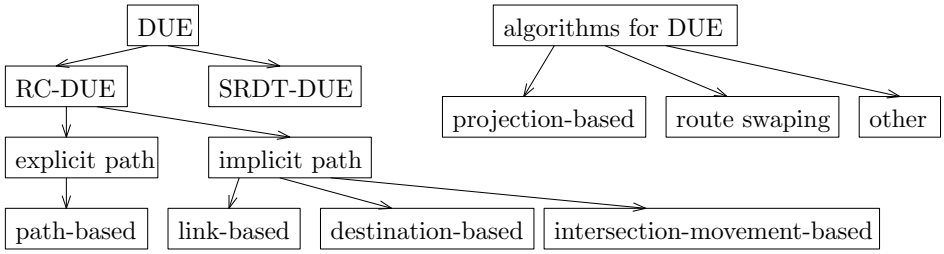


Figure 3.2: Categorization of the DUE formulations and algorithms

Early works that deal with explicit event-based methods (Wu and Liu, 2011) track shockwaves in space and time. The extension of these methods to the network is very difficult (Raadsen and Bliemer, 2019). The implicit event-based algorithms track shortwaves only on link boundaries. The idea of the implicit event-based method was used by Raadsen et al. (2016). This algorithm removes the disadvantages of discretization.

- *The event-based solution with a non-triangular FD* was published by Raadsen and Bliemer (2019) and Bliemer and Raadsen (2019). It turns out that the use of GLTM has the positive influence on the smoothness of convergence to the DUE. On the other hand, the implementation is difficult.

The *metamodeling* is the next technique how to speed up the DNL procedure. The metamodel is the 'model of the model'. The aim is to create an approximation model that improves the computational performance of the DNL. A representative of this method is the statistical metamodeling by Song et al. (2017) that uses the method known as *Kriging*. The disadvantage is a huge preprocessing but after that the query is fast.

3.2.2 The formulations and solutions of dynamic user equilibrium

The dynamic user equilibrium (DUE) is the state of the network, where there is no faster alternative (path) for any user at any time to the destination. This model assumes that all users choose the fastest alternative. The user equilibrium model was published by Wardrop (1952) and it is called Wardrop's first principle. In Figure 3.2 there is a categorization of the DUE formulations and algorithms for solution.

There are two basic types of the dynamic equilibrium: route-choice (RC) DUE and simultaneous route-and-departure-time (SRDT) DUE by Friesz et al. (1993). The RC-DUE model is the simplest dynamic extension of the traffic assignment problem (TAP) and solves only the user route choice. Moreover, the SRDT-DUE model selects the optimal departure time.

There are two basic groups of RC-DUE formulations: with explicit path and with implicit path enumeration. The first group is represented by a classical path-based formulation. This approach divides the flow into the paths from the origin to the destination. Most of the path-based formulation needs to know the set of the used paths for every origin-destination pair before the computation and needs to store them. This is the main disadvantage of this approach. The second disadvantage is the fact that the paths in the path set are overlapped a lot and therefore the paths influence one another. This leads to problems in reaching the equilibrium.

The second group includes the link-based (Ran et al., 1996), intersection-movement-based (Long et al., 2013) and destination-based formulation (Gentile, 2016), (Himpe, 2016). These three methods are very similar and have the same idea. They do not store the paths so they are less memory demanding, on the other hand, they store the distribution matrix for every node. The shortest path search is needed.

The algorithms that solve the DUE can be split into three categories.

- *The projection-based algorithms* are derived from the basic projection algorithm (Facchinei and Pang, 2004) for solving monotone Variational Inequalities (VI) or from Gradient Projection (GP) methods. It should be noted that the delay operator does not have to be monotone and continuous due to spillback, therefore, the convergence is not guaranteed. This approach is most commonly used and provides reasonable performance. The convergence is not too smooth. The representatives are: the fixed-point algorithm by Friesz et al. (2011), alternating direction by Lo and Szeto (2002), descent method by Han and Lo (2004) and Szeto and Lo (2004), extragradient algorithm by Long et al. (2013), and project methods (Jang et al., 2005; Ukkusuri et al., 2012; Gentile, 2016).

- *The route swapping algorithms* shift the flow from more expensive paths to cheaper paths. The approach is very similar to the projection-based algorithms and the performance is also very similar (Kolovský et al., 2019c). The representatives are (Huang and Lam, 2002), (Tian et al., 2012) and (Szeto and Lo, 2006).

- The last two algorithms, *self-adaptive* and *proximal-point*, were used by Han et al. (2015). These algorithms provide smoother convergence but the performance (convergence rate) is very poor, therefore, they are not usable in practice.

A lot of mentioned algorithms is formulated for SRTD-DUE but the RC-DUE is a special case of SRDT-DUE, so the algorithms can be easily adjusted. In this chapter, the Quasi Gradient projection by Gentile (2016) was used because all the methods provide very similar performance.

3.3 Main concept of the proposed solution

In this chapter, the Rasdsen's event-based GLTM is used as a DNL (Raadsen and Bliemer, 2019) because this method provides a continuous-time solution and the precision of the DNL can be driven easily. More precisely, the input is PWC time-dependent ODM and the output is the PWL travel time function for every edge in the graph. The algorithm expects that every flow in the network is PWC. As equilibrium, the classical RC-DUE is used because it is used widely in practice and the computation is much faster. In the end the RC-DUE is a special case of the SRDT-DUE. In this chapter, as the equilibrium formulation, the intersection-movement-based formulation is used because it does not need to store the paths.

In Section 3.3.1 the definitions are precisely introduced. In Section 3.3.2 the whole proposed solution is described.

3.3.1 Definitions

Let $G = (V, E)$ is a directed graph, where V is a set of nodes (vertices) and $E \subset \{(i, j) : i, j \in V\}$ is a set of directed edges. This graph represents a road network. $Z = \{1, 2, \dots |Z|\} \subset \mathbb{Z}$ is a set of zones that represents the places where the vehicles enter the graph and leave the graph.

The route-choice (RC) DUE model is used in this chapter. The input is time-dependent origin-destination matrix (ODM), defined as

$$\text{ODM} = \{h^{ij}(t) : \forall (i, j) \in W\} \tag{3.1}$$

where $h^{ij}(t)$ is the flow of vehicles (departure rate) between the zone $i \in Z$ and the zone $j \in Z$ at the time t and $W = Z \times Z = \{(1, 1), (1, 2), \dots, (2, 1), (2, 2), \dots, (|Z|, |Z|)\}$ is a set of origin-destination pairs. The ODM must meet conditions

$$\int_{t_{min}}^{t_{max}} h^{ij}(t)dt = Q_{ij} \tag{3.2}$$

$$h^{ij} \geq 0 \tag{3.3}$$

where Q_{ij} is the total number of vehicles corresponding with the OD pair $(i, j) \in W$ and $[t_{min}, t_{max}]$ represents the simulation time interval.

The simplified intersection-movement-based DUE formulation by Long et al. (2013) is described here. The set of all incoming edges to the node n is denoted $I_n \subset E$ and the set of outgoing edges $J_n \subset E$, then the basic notations are:

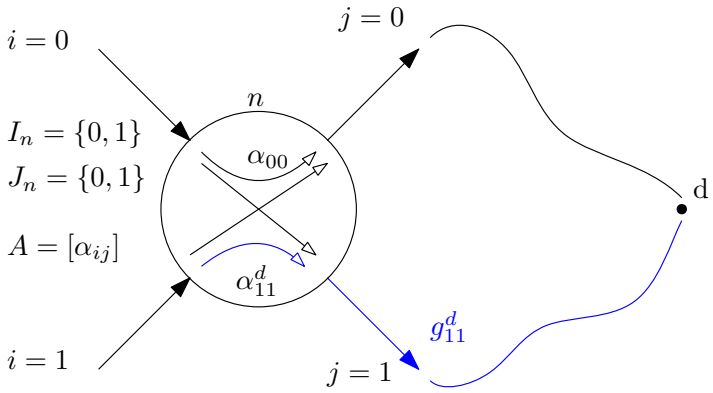


Figure 3.3: Node distribution notation

$\alpha_{ij}(t)$	the proportion of flows that leave the edge $i \in I_n$ and want to continue to the edge $j \in J_n$
$\alpha^d_{ij}(t)$	the proportion of flows that leave the edge $i \in I_n$, want to continue to the edge $j \in J_n$ and travel to the destination $d \in Z$
$m^d_i(t)$	the proportion of flows that leave the edge $i \in I_n$ and travel to the destination $d \in Z$
$m^d_j(t)$	the proportion of flows that enter the edge $j \in J_n$ and travel to the destination $d \in Z$
$g^d_i(t)$	the minimum travel time from the target node of the edge $i \in I_n$ to the destination zone d
$g^d_{ij}(t)$	the minimum travel time from the target node of the edge i to the destination zone d using edge $j \in J$
$A = [\alpha_{ij}]$	node distribution matrix
$A^d = [\alpha^d_{ij}]$	node distribution matrix for the destination zone d
$\boldsymbol{\alpha} = [A^d]$	vector of proportions, the result of RC-DUE

In Figure 3.3 there is an example of an intersection with two incoming edges $I_n = \{0, 1\}$ and two outgoing edges $J_n = \{0, 1\}$. There are also all intersection movements that connect the incoming and outgoing edges (arrows inside the circle). The distribution matrix A is created by the proportions of flow α_{00}, α_{01}, α_{10} and α_{11} that are related to the intersection movements. The blue color represents the fastest path g^d_{11} from the target node of the incoming edge $i = 1$ to the destination d through the outgoing edge $j = 1$.

One element of the distribution matrix of the node $n \in V$ can be calculated as

$$\alpha_{ij} = \sum_{d \in D} \alpha^d_{ij} m^d_i \tag{3.4}$$

where $D \subset Z$ is a set of destinations. The distribution matrices and flow proportions

have to meet the following equations

$$\sum_{j \in J_n} \alpha_{ij} = 1 \tag{3.5}$$

$$\sum_{j \in J_n} \alpha_{ij}^d = 1 \tag{3.6}$$

$$\sum_{d \in D} m_i^d(t) = 1 \tag{3.7}$$

$$\sum_{d \in D} m_j^d(t) = 1 \tag{3.8}$$

The sum of proportions in one row of the proportion matrices must be 1. The sum of proportions of the flows entering or leaving the edge must be also 1. These equations form the feasible set Λ for the proportion vector $\boldsymbol{\alpha}$. The feasible set is a set of all possible proportion vectors.

According to Long et al. (2013), the user equilibrium occurs if

$$g_{ij}^d(t) \begin{cases} = g_i^d(t) & \text{if } \alpha_{ij}^d(t) > 0 \\ \geq g_i^d(t) & \text{if } \alpha_{ij}^d(t) = 0 \end{cases} \tag{3.9}$$

It follows that if the outgoing edge j is used by users to travel to the destination $\alpha_{ij}^d > 0$ then the travel time through this edge $g_{ij}^d(t)$ must be equal to the minimal travel time to the destination g_i^d, otherwise it is higher than the minimal travel time g_i^d to the destination d. This must apply to all times t.

One of the options how to define and formulate the optimization problem of DUE is to use the variational inequality (VI) problem. The vector $\boldsymbol{\alpha}^*$ represents the optimal solution that satisfies equation (3.9). The VI for RC-DUE is (Long et al., 2013), (Gentile, 2016)

$$\sum_{n \in N} \sum_{d \in D} \sum_{i \in I_n} \sum_{j \in J_n} \int_{t_{min}}^{t_{max}} g_{ij}^{d*}(t) \left(\alpha_{ij}^d - \alpha_{ij}^{d*} \right) dt \geq 0 \quad \forall \boldsymbol{\alpha} \in \Lambda \tag{3.10}$$

where $D \subset Z$ is the set of destination zones, $I_n \subset E$ is the set of incoming edges and $J_n \subset E$ is the set of outgoing edges for the node $n \in V$. If the inequality (3.10) is met for all possible proportion vectors $\boldsymbol{\alpha} \in \Lambda$ then the proportion vector $\boldsymbol{\alpha}^*$ is the optimal solution that also satisfies (3.9) so the aim is to find this optimal vector.

Let $f(t)$ is a piecewise linear (PWL) function. The number of linear pieces of f is $N(f)$. Let $L(g)$ be a set of definition intervals of PWL function g. The set is

$$L(g) = \left\{ \left[t_{lo}^1, t_{hi}^1 \right), \left[t_{lo}^2, t_{hi}^2 \right), \ldots, \left[t_{lo}^{N(g)}, t_{hi}^{N(g)} \right] \right\} \tag{3.11}$$

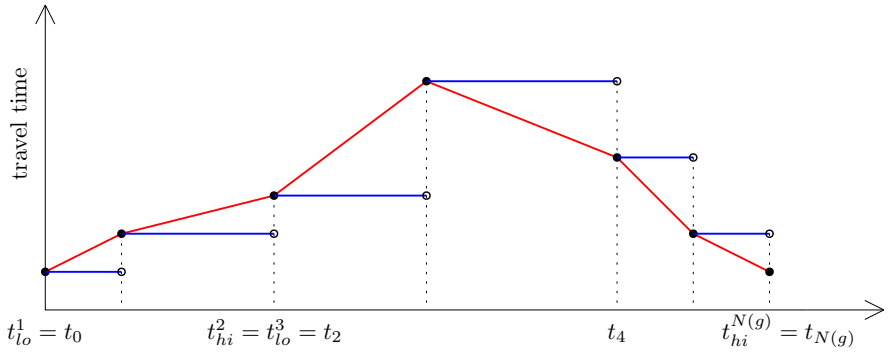


Figure 3.4: Example of PWC function (blue) and PWL function (red)

where $t^{l-1}_{hi} = t^l_{lo}$ subject to $2 \leq l \leq |L(g)|$ and $N(g)$ is the number of linear pieces of the function g.

Let the set of *breakpoints* of the function g is

$$T(g) = \{t_{lo} : [t_{lo}, t_{hi}) \in L(g)\} \cup \left\{t^{N(g)}_{hi}\right\} = \{t_0, t_1, t_2, \ldots, t_{N(g)}\} \tag{3.12}$$

It follows that $|T(g)| - 1 = |L(G)| = N(g)$. In Figure 3.4 there is an example of the PWL function (red color).

Let $H = \{h_1, h_2, \ldots\}$ is the set of PWL functions, then

$$T\left(\sum_i h_i\right) = \bigcup_i T(h_i) \tag{3.13}$$

It means that the set of breakpoints of a sum of functions is equal to the union of the set of breakpoints of functions. It follows that

$$\max_i N(h_i) \leq N\left(\sum_i h_i\right) \leq \sum_i N(h_i) \tag{3.14}$$

It means that the number of breakpoints of the result can be up to the sum of all breakpoints in the set H.

Now, let $H = \{h_1, h_2, \ldots\}$ be the set of piecewise constant (PWC) functions. In Figure 3.4 there is an example of the PWC function (blue color). Everything that has been written about a PWL function applies to the PWC function. Moreover, the following equation holds:

$$T\left(\prod_i h_i\right) = \bigcup_i T(h_i) \tag{3.15}$$

This equation has the same meaning as (3.13). Only the sum is replaced by the multiplication.

Table 3.1 summarizes the results of the operations in the cases when the function is PWL or PWC function. Only the results of multiplication for PWL functions is not a PWL function.

operation	f is PWC	f is PWL
$f_1 \pm f_2$	PWC	PWL
$f_1 * f_2$	PWC	is not PWL
$f_1(f_2)$	PWC	PWL
$f_1 * c$	PWC	PWL
$f_1 \pm c$	PWC	PWL

Table 3.1: Results of operations ($c \in \mathbb{R}$)

3.3.2 Solution scheme

The main idea of the proposed solution is to use the PWL and PWC functions instead of a grid (time discretion). The precision of the solution is set by approximation parameters as a maximal relative or absolute error of the function instead of the size of the time step. So the following steps have to be developed as continuous-time (PWL or PWC):

- Dynamic network loading (DNL) - The event-based generalized link transmission model (E-GLTM) by Raadsen and Bliemer (2019) and Bliemer and Raadsen (2019) was used because it provides the continuous-time solution. The level of approximation is set by a flow threshold δ_q and mixture propagation threshold δ_m. The loading of the network can be performed in the exact form (except simplified fanning). This part is described in Section 3.4.

- Time-dependent shortest path search (profile search) - We used the modified version of ε-LCA-BS that was presented in Chapter 2 because it provides an effective ε-approximation of the TDSP. The maximal relative error ε_s drives the level of approximation. The exact result is also supported. This method needs to be adjusted so that it computes the all-to-one problem. In Section 3.5 the all-to-one modification of the ε-LCA-BS is described.

- Node distribution update (flow proportion update) - In this step, the proportion of the flow α_{ij}^d (node distribution) is updated using minimal travel time function to the destination $g_{ij}^d(t)$. This update must be constructed so that it respects the PWL input and is continuous-time. Here, the approximation of TTFs by PWC functions is necessary because the node distributions are PWC. It follows that this step cannot be exact. The maximal relative ε_g or absolute error δ_g is kept.

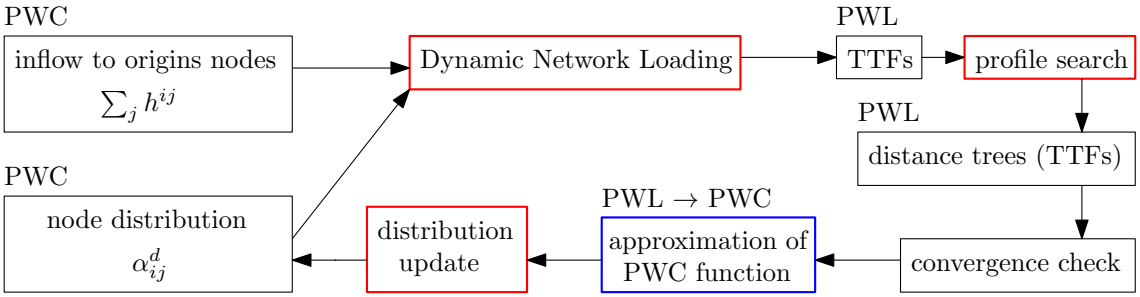


Figure 3.5: The general solution scheme

After the update, the node distribution is simplified with the maximal absolute proportion error δ_p. This procedure is described in Section 3.6.

In Figure 3.5 there is the solution scheme for DUE. The red color represents the procedures that loss the information but do not have to. It means that the maximum allowed error can be set to the zero. The blue color represents the step where the approximation parameters cannot be zero. Here some information must be lost. In this case it is the approximation of the PWL function by a PWC function.

3.4 Dynamic network loading

This section contains a description of the used DNL method. The event-based general link transmission model by Raadsen and Bliemer (2019) was used. This model use the dual-quadratic fundamental diagram that is defined by five parameters that are: capacity q^{max}, maximum jam density k^{max}, maximum speed γ^{max}, minimum backward wave speed γ^{min}, and critical speed γ^{crit}.

This model had to be extended by a node events that represent the changes in node distribution, due to the intersection-movement-based formulation of RC-DUE. A new approximation approach that improves precision is proposed.

In this section, the original method by Raadsen is described with high abstraction. More details can be found in the original paper.

3.4.1 Extension

Due to intersection-movement-based DUE formulation, the changes in the node distribution α_{ij} has to be take into account during the simulation. Therefore, in this section we propose a modification which reflects these changes in the node distribution.

The whole loading procedure by Raadsen and Bliemer (2019) is designed as an event-based simulation. Every change in the network is implemented as an event. If the inflow to the edge is changed, a new constant piece is added to the inflow function and

a new event that ensures the propagation of the change to the target node of the edge is released. This event is called a *forward event*. The time, when the change (event) reaches the target node, is estimated and it is called a *predicted time*. If the predicted time of the event is equal to the time of the simulation, the flow demand in the target node of the edge is changed. It follows that the inputs to the node were also changed and thus the node model must be recomputed. The node model triggers the changes of the inflow to the outgoing edges and the outflow of the incoming edges. The change of the outflow from the incoming edge triggers a *backward event* that implements the backward kinematic wave that is propagated in the opposite direction than the forward event and it carries the information about the change of the flow supply at the source node of the edge.

For DNL, it is also important to track the proportion of the flow related to the path or the destination (in our case m_i^d and m_j^d). These proportions are called a mixture (e.g., 30% of flow go to the destination 1 and 70% of flow go to the destination 2). If the mixture at the source node of the edge is changed, the event that represents this change is released and it is called a *mixture event*. As for the forward and backward event, the predicted time is estimated and the event is propagated through the edge to the target node. The change of the mixture at the target node also influences the node model.

As mentioned above, there are three basic types of event: forward, backward and mixture. This original procedure assumes that the distribution (matrix A) is not driven by parameters of node model. But in our case the node distribution matrix is time-dependent because

$$A = A(t) = [\alpha_{ij}(t)] \tag{3.16}$$

Therefore, in comparison with the original algorithm, a *node event* that represents the change of the distribution matrix, was added. The predicted time of the node event is the time of distribution change. The event is created for every breakpoint in the proportion functions related to the node $n \in V$. The set of breakpoints for the node n is

$$T_n = \bigcup_{i \in I_n, j \in J_n, d \in D} T(\alpha_{ij}^d) \tag{3.17}$$

where I_n is a set of incoming edges, J_n is a set of outgoing edges and $D \subset Z$ is a set of destination zones. The predicted time of the event is set to the time of the breakpoint.

To make the algorithm simpler, an *origin event*, which implements the change of the inflow to the network and change of the mixture at the origin node, was also added. The inflow to the origin node $i \in V$ is

$$h^i = \sum_{j \in D} h^{ij} \tag{3.18}$$

and, according to (3.13), the origin event should be created for every breakpoint in the

inflow function. The set of breakpoints for the origin $i \in Z$ is

$$T^i = \bigcup_{j \in D} T(h^{ij}) \tag{3.19}$$

The predicted time of the origin event is also set to the time of the breakpoint.

Algorithm 4: Adjusted E-GLTM algorithm

```
1  PQ = minimum priority queue of events where key is predicted time
2  ∀n ∈ N : add all node events related to node n to PQ according to (3.17)
3  ∀o ∈ O : add all origin events related to origin o to PQ according to (3.19)
4  t = 0
5  while PQ is not empty and (t < t_max) do
6      e = remove the event with the earliest predicted time from PQ
7      t = predicted time of event e
8      if e is forward event then
9          process the event and change the demand at the end of the edge
10         recompute the model of the target node
11     else if e is backward event then
12         process the event and change the supply at the beginning of the edge
13         recompute the model of the source node
14     else if e is mixture event then
15         process the event and change mixture at the end of the edge
16         recompute the model of the target node
17     else if e is node event then
18         recompute the node model corresponding with the node event
19     else if e is origin event then
20         change demand and mixture at the origin
21         recompute the node model corresponding with the origin event
22     Inflow to the outgoing edges is changed and forward events are released;
23     Outflow from the incoming edges is changed and backward events are
         released;
24     Mixture at the beginning of the outgoing edges is changed and the mixture
         events are released;
```

The adjusted loading procedure that used in this chapter is shown in Algorithm 4. The key element of the algorithm is a priority queue (PQ) of the events where the key is the predicted time of the event. In the initialization part (line 1-4), all origin and node events are added to the PQ and the simulation time t is set to zero. The main loop

is repeated as long as the simulation time t is smaller than the maximum simulation time t_{max} or the PQ is not empty. First, the event e with the earliest predicted time is removed from the PQ and the simulation time t is set to this predicted time. The event is processed according to its type and the corresponding node model is recomputed. In the end, the inflow to the outgoing edges, the outflow from the incoming edges, and mixture at the beginning of the outgoing edges are changed and the forward, backward, and mixture events are added to the PQ.

3.4.2 Approximation

As mentioned above, every event modifies the inputs to the node model and thus triggers the creation of new events that also generate additional events. The number of events in the simulation highly increases and this increases computational time. It follows that the approximation of the exact solution is needed. The basic idea is that all events are not equally important so some of these can be thrown out (discarded).

The original method by Raadsen and Bliemer (2019) uses the approach based on a *flow threshold* δ_q. It means that if the size of flow change, which is called a *flow step* Δ_q, is smaller than the threshold, the flow step and the corresponding event are thrown out. This method has a problem with small flow steps that are valid for a long time (e.g., flow change 1 veh/h valid for 1 hour). This small long step is thrown out but it would be valid for a long time so the total error in the number of vehicles is quite big. The method has the same problem with very short steps. The step has very small influence but it is propagated further and may cause numerical problems.

Our weighted flow threshold method tries to eliminate these disadvantages. The main idea is very simple. The flow step is weighted by the time length of the step. This idea is presented in Figure 3.6. The thick green line represents the exact departure rate (flow) and the color dashed lines are the simplified departure rates. The red is the result of flow threshold simplification by Raadsen. The blue color is the product of weighted flow threshold simplification. As you can see, the light blue error areas are smaller than the light red areas.

The main problem of the proposed method is that the time length of the step is not available in every case because the future state of the simulation is not known. The flow PWC function is known only up to the current simulation time t. Another approximate approach is proposed to solve this issue. The main idea is to find the next scheduled event that can influence the departure flow (flow function). In the case of node or origin event, the next scheduled event can be found simply because all the events are known before the simulation starts. In other cases, with high likelihood, the next scheduled event will be connected to some income or outcome edge. If there is no event on neighboring edges, the time length of the step is estimated as the minimum travel time of these edges (or actual travel time).

The estimated time length of the step is used as the weight for the flow step. The

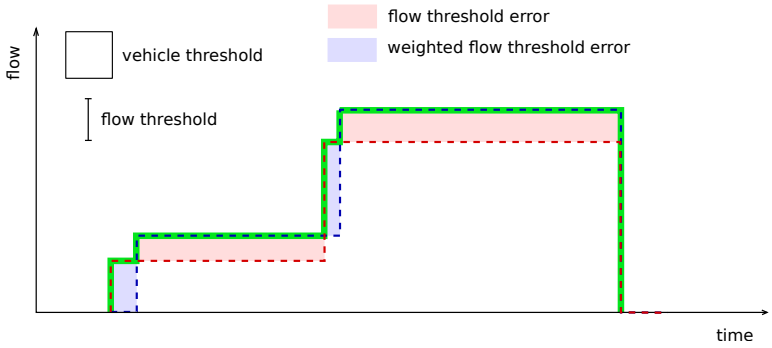


Figure 3.6: Comparison of two types of threshold

flow step is propagated if

$$\delta_q^w < \Delta_q * \Delta_t \tag{3.20}$$

where δ_q^w is the weighted flow threshold, Δ_q is the flow step, and Δ_t represents the time length of the step. This proposed method of weighted flow threshold also eliminates the fluctuations of the precision of the TTFs. In the same way, the propagation of the mixture can be approximated. The change of the mixture

$$\Delta_m = \max_{d \in D} |m_j^d(t) - m_j^d(t-1)| \tag{3.21}$$

is also weighted by the time length of the step. The $m_j^d(t)$ is the mixture in the current simulation time and $m_j^d(t-1)$ represents the previous mixture. The mixture is propagated if

$$\delta_m^w < \Delta_m * \Delta_t \tag{3.22}$$

where δ_m^w is a weighted mixture threshold.

3.5 The continuous-time all-to-one shortest path problem

The information about the travel time is necessary for finding the equilibrium in the road network. The time-dependent shortest path searches are performed in every iteration of the proposed solution after the DNL. Based on this travel time the information about route choice is updated.

In our case, the RC-DUE is formulated as intersection-movement-based (see to Section 3.3.1). Therefore, the PWL time-dependent shortest path search for all departure times (profile search) is necessary for providing the continuous travel time functions g_{ij}^d. Specifically, the minimal travel time function from all nodes to the destination is needed. This variant is called the all-to-one earliest arrival (EA) time problem.

In this Section, the g represents the travel time function (TTF) and f is a arrival function (AF). As was defined in Chapter 2, the relationship between the functions are

$$f(t) = g(t) + t \tag{3.23}$$

As the basic algorithm, the ε-LCA-BS, which was modified so that it solves the all-to-one EA problem, was used, because the original algorithm solves the one-to-all problem.

First, the exact version of the all-to-one EA Label Correcting Algorithm (LCA) is presented, because ε-LCA-BS is only an approximation version of LCA. The changes compared to the classical LCA by Orda and Rom (1990) are:

- The searching starts at the destination node and uses the reverse graph $G_r = (V, \{(j,i) : (i,j) \in E\})$.
- The relaxation of the original edge $(u,v) \in E$ is performed using the following equation

  $$f_u = \min(f_u, f_v(f_{uv})) \tag{3.24}$$

  where f_u is the arrival function to the destination node from the node u and f_{uv} is the arrival function to the node v from the node u (so called the *edge arrival function*).

The ε-approximation of the all-to-one EA LCA is implemented in the same way as in ε-LCA-BS. The approximation helps us maintain a reasonable number of linear pieces in the resulting TTFs. The maximal relative error ε_s for the searching should be set to a smaller value than the required relative gap of the RC-DUE.

3.6 The node distribution update

The continuous-time update of the node distribution is an important part of the whole solution. The aim is to update the node distributions α_{ij}^d using the travel time functions from the current node to the destination so that the solution will converge to the RC-DUE. This update should be also continuous-time but the input TTFs are PWL and the proportion functions α_{ij}^d are PWC, so some approximation of PWL function by PWC must be performed here.

The algorithm that updates the node distribution α_{ij}^d is designed so that various updating method that are known from the state-of-the-art, can be used (e.g., fix-point method, gradient methods, route-swapping method), see Algorithm 5.

First, all-to-one ε-LCA-BS to a destination $d \in D$ is computed with maximum relative error ε_s and than for all $n \in V$ the proportion of flow α_{ij}^d that travel to

destination d is updated. The functions g_{ij}^d have to be determined according to following equation

$$g_{ij}^d(t) = f_{kd}(f_j(t)) - t \tag{3.25}$$

where $k \in V$ is the target node of the edge $j \in J_n$, f_j is the edge arrival function of the edge j and f_{kd} represents the arrival function from node k to destination d computed by ε-LCA-BS. At this point, the function g_{ij}^d can be also simplified according to the mechanism that is used in the ε-LCA-BS and the maximal relative error ε_s of the travel time function will be still maintained. Now, the functions g_{ij}^d are still PWL. The proportion of flow α_{ij}^d is PWC so the g_{ij}^d must be approximated by PWC with maximum absolute error δ_g or with maximum relative error ε_g. This operation can be performed optimally by the method according to Konno and Kuno (1988).

Algorithm 5: The update procedure

```
for all d ∈ D do
    perform all-to-one ε-LCA-BS
    for all n ∈ N do
        for all i ∈ I_n do
            ∀j ∈ J_n : compute g_ij^d according to (3.25)
            approximate g_ij^d by PWC function with maximum absolute error δ_g
              or relative error ε_g
            for all t ∈ ⋃_{j∈J_n} T(α_ij^d) ∪ ⋃_j T(g_ij^d) in chronological order do
                update α_ij^d(t) by suitable method (e.g., QG method)
                adjust (project) α_ij^d(t) so that Σ_j α_ij^d(t) = 1
            simplify α_ij^d with maximum absolute error δ_p
```

Now both functions are PWC, so the update of the proportion can be performed. According to the (3.13), the update must be performed for every breakpoint of the involved functions. The set of breakpoints for which the update have to be applied is

$$\bigcup_{j \in J_n} T(\alpha_{ij}^d) \cup T(g_{ij}^d) \tag{3.26}$$

In this chapter the Quasi gradient (QG) method by Gentile (2016) is used for the updating. Let $B_i^d \subset J_n$ be a set of outgoing edges with non zero probability α_{ij}^d connected with destination d and incoming edge i. According to QG, the distribution shifts are:

$$\Delta\alpha_{ij}^d(t) = \frac{g_{avg}^d(t) - g_{ij}^d(t)}{\gamma_{ij}^d(t)}, \quad \forall j \in B_i^d \tag{3.27}$$

$$\Delta\alpha_{ij}^{d}(t) = 0, \quad \forall j \in J_n \setminus B_i^d \tag{3.28}$$

where

$$g_{avg}^{d}(t) = \frac{\sum_{j \in B_d} g_{ij}^{d}(t)}{|B_d|} \tag{3.29}$$

and $\gamma_{ij}^{d}(t)$ represents a gradient of travel time but this gradient cannot be determined easily so in this chapter the value was set globally as $\gamma_{ij}^{d}(t) = \gamma$. The γ can be understood as a step. The set B_i^d must be determined iteratively by *Greedy algorithm*. See Gentile (2016). Finally, the distribution is updated as

$$\alpha_{ij}^{d}(t) = \alpha_{ij}^{d}(t) + \beta_i^d \Delta\alpha_{ij}^{d}(t) \tag{3.30}$$

where β_i^d is shorter coefficient maintaining that $\alpha_{ij}^{d}(t) \geq 0$ and can be computed as

$$\beta_i^d = \min\left(1, -\frac{\alpha_{ij}^{d}(t)}{\Delta\alpha_{ij}^{d}(t)}, \forall j \in B_d : \Delta\alpha_{ij}^{d}(t) < 0\right) \tag{3.31}$$

According to (3.6), the sum of probabilities must be one so the updated node distributions from the previous step must be projected to feasible set Λ. The projection is

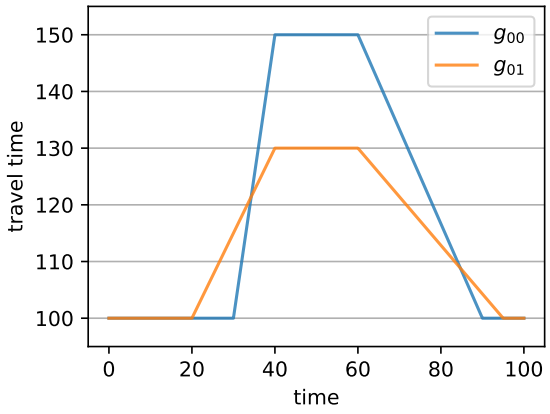
$$\alpha_{ij}^{d} = \alpha_{ij}^{d} + k, \quad \forall j \in B_i^d \tag{3.32}$$

where $k = \frac{1-\sum_j \alpha_{ij}^{d}}{|B_i^d|}$.

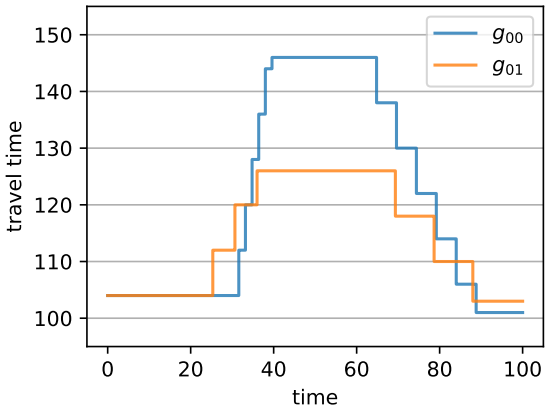
In the end the updated PWC proportions α_{ij}^{d} are simplified with the maximum absolute error δ_p. This simplification can be also performed optimally in linear time by Konno and Kuno (1988). The simplification must also respect the condition (3.6).

The approximation of the g_{ij}^{d} by PWC function is the only step where the maximum error cannot be set to zero. The other approximation parameters δ_q and ε_s can be zero.

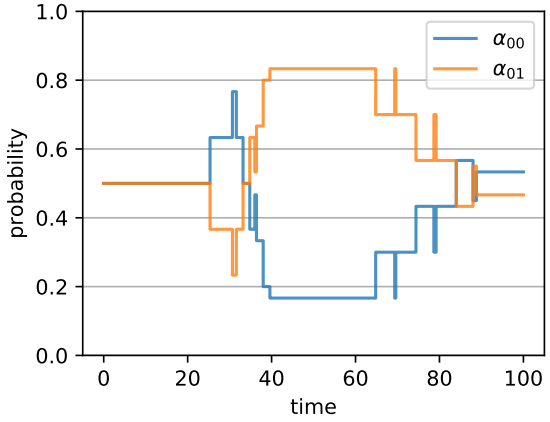
In Figure 3.7 you can see example of distribution update. In this case, there are two alternatives. First is represented by TTF g_{00} and by probabilities α_{00}. The second alternative is represented by TTF g_{01} and by probabilities α_{01}. In Figure 3.7a there are the TTFs for both alternatives as was obtained from DNL. Both TTFs have five linear pieces (six breakpoints). Figure 3.7b shows the TTFs approximated by PWC functions with maximal absolute error $\delta_g = 4.0$. Now the TTF g_{00} has 13 constant parts (14 breakpoints) and the g_{01} has 7 parts (8 breakpoints). Initially, the node distribution is set to 0.5 for the whole simulation interval. According to (3.26), for every breakpoint from g_{00}, g_{01}, α_{00}, and α_{01}, the QG projection was applied. Figure 3.7c shows these result functions. The γ was set to 30.0. According to (3.14), the updated distribution PWC function have up to 21 constant pieces. The last step is simplification of probabilistic functions α_{00} and α_{01} with the maximal absolute error $\delta_p = 0.1$. The final results is in Figure 3.7d. Both final probability functions have 6 constant pieces (7 breakpoints).



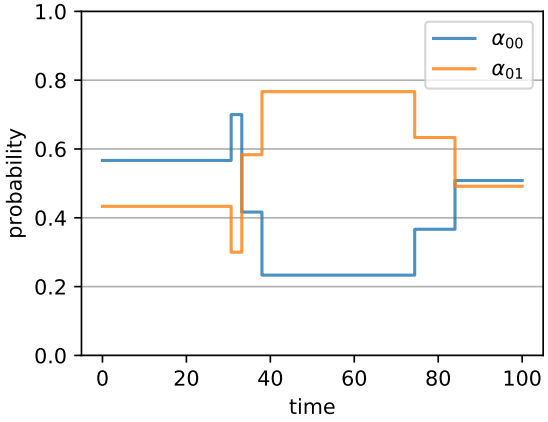
(a) PWL travel time functions



(b) approx. of TTFs by PWC func. $\delta_g = 4.0$



(c) node distribution after update by QG



(d) node distr. after simplification $\delta_p = 0.1$

Figure 3.7: Example of node distribution update

3.6.1 Approximation parameters

As mentioned above, the precision of the solution and thus the speed of calculation is driven by five parameters:

- The flow threshold δ_q or the weighted flow threshold δ_q^w for DNL,
- the mixture threshold δ_m or the weighted flow threshold δ_m^w for DNL,
- the maximum relative error ε_s for the continuous-time TDSP,
- the maximal absolute δ_g or relative ε_g error during approximation PWL functions by PWC functions,

- the maximal absolute error δ_p for simplification of the node distribution α_{ij}^d.

These parameters should be coordinated. For example, if the DNL is precise, the relative absolute error ε_s for the shortest path search must be also set to a small value, otherwise the DNL accuracy would be useless. The coordination process is not straightforward so for the purposes of this chapter, the parameters are coordinated empirically. Automatic smart coordination can be part of future research.

3.7 Numerical tests

The whole solution was written in Scala programming language and has more than 11500 lines of code. The tests were performed on a laptop with 16 GB of memory and with four-core processor (Intel(R) Core(TM) i5-8250U CPU @1.60GHz). One thread was used only.

For the testing, three data sets were used: Twin network, Braess network, and the real traffic models of Anaheim and Eastern Massachusetts by Transportation Networks for Research Core Team (2021).

For measuring, how close the solution is to the exact equilibrium, the definition of a relative gap is necessary. The average relative gap in this section is understood as

$$\chi = \frac{\sum_{n\in N}\sum_{i\in I_n}\sum_{d\in D}\chi_{ni}^d}{|D|\sum_n |I_n|} \tag{3.33}$$

where

$$\chi_{ni}^d = \frac{\int_{t_0}^{t_{max}} \overline{g} - g_i^d}{(t_{max}-t_0)\overline{g}_{max}} \tag{3.34}$$

$$\overline{g} = \max_{j\in J_n} g_{ij}^d \tag{3.35}$$

$$\overline{g}_{max} = \max_{t\in[t_0,t_{max}]} \overline{g}(t) \tag{3.36}$$

Let us recall that g_i^d is minimum travel time from the target node of the edge i to the destination d and $g_{ij}^d(t)$ is minimum travel time from the target node of the edge i to the destination zone d using edge $j \in J$.

This formulation assumes that all paths to every destination are used. In case that some path from edge i to destination d using edge j is not used, this path must be eliminated from the computation.

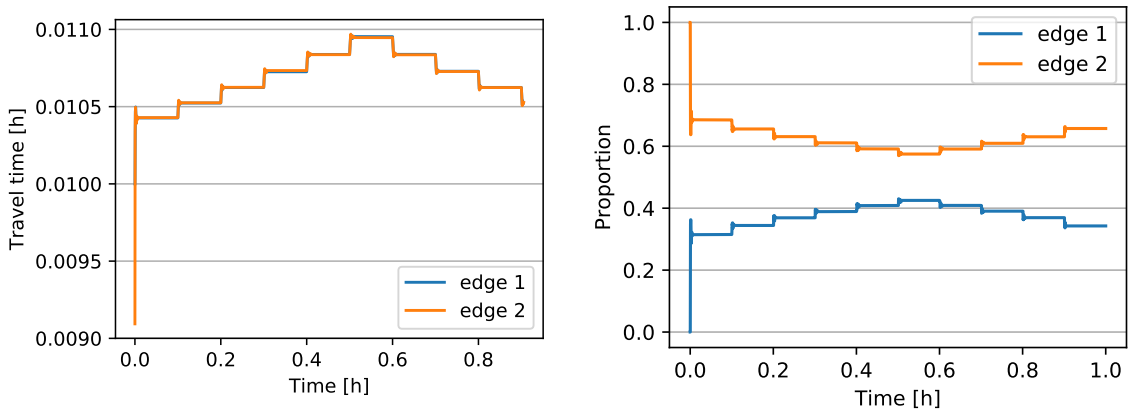
3.7.1 The Twins

The Twin network consists of two parallel edges with different fundamental diagrams. In Table 3.2 there are parameters of the edges. The inflow starts at 1000 vehicles per

edge	γ^{max} [km/h]	q^{max} [veh/h]	k^{max} [veh/km]	γ^{crit} [km/h]	γ^{min} [km/h]
1	100	2000	222	52	-17
2	110	1500	166	63	-17

Table 3.2: Parameters of the edges in Twin network

hour (veh/h) and every 6 minutes is increased by 100 to 1500 veh/h and after that every 6 minutes is decreased by 100 veh/h. The simulation time was set to 1.2 hours. In Figure 3.8 you can see the resulting travel time functions and the proportional functions. The average relative gap $3 \cdot 10^{-4}$ was achieved using the following settings: $\delta_p = 10^{-4}$, $\delta_q = 0.49$ veh/h. As you can see, the travel time functions almost overlap. That is the result we wanted to achieve. This simple test was the aim to confirm that the solution scheme can work. This reached accuracy is perfectly sufficient for practical use.



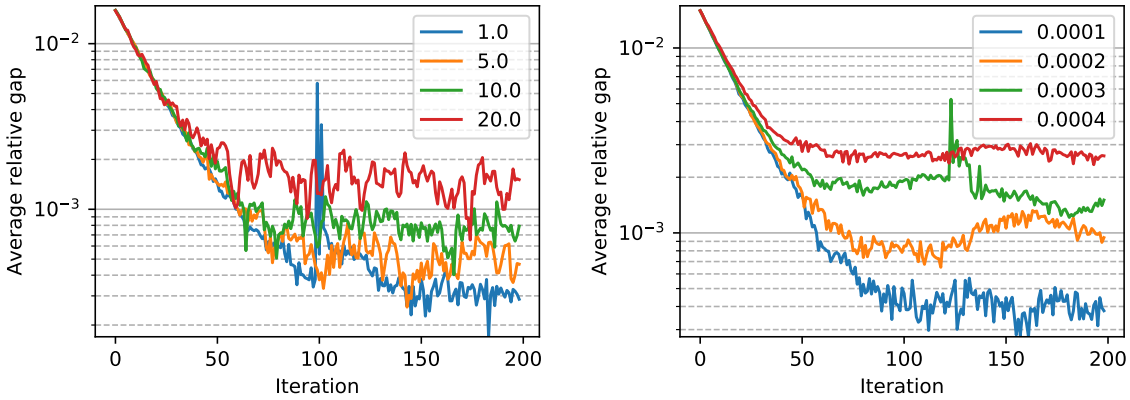
(a) travel time functions

(b) proportion functions

Figure 3.8: Results for twin network

Also for testing the approximation parameters this network was used. First, the dependence of reached average gap to the approximation level of DNL was examined. For testing purposes, the flow threshold was set to 0.49 multiple of the fan step. So if the fan step is 10 veh/h, the flow threshold is set to 4.9 veh/h. The fan step was set to 1, 5, 10 and 20 veh/h. In Figure 3.9a there are the convergence charts for each setup. As you can see, the reached relative gap is highly dependent on the flow threshold. The reached relative gaps are $3 \cdot 10^{-4}$, $5 \cdot 10^{-4}$, $8 \cdot 10^{-4}$ and $14 \cdot 10^{-4}$.

The second test measures the dependence of the reached relative gap on maximal absolute error δ_p. The δ_p was set to $1 \cdot 10^{-4}$, $2 \cdot 10^{-4}$, $3 \cdot 10^{-4}$ and $4 \cdot 10^{-4}$. The $\delta_q = 0.98$ and fan step was set to 2.0. In Figure 3.9b there are the convergence curves. The reached relative gaps are $4 \cdot 10^{-4}$, $10 \cdot 10^{-4}$, $14 \cdot 10^{-4}$ and $25 \cdot 10^{-4}$. The conclusion is



(a) various settings of δ_q and fan step

(b) various settings of δ_p

Figure 3.9: Dependence of convergence on approximation for twin network

the same as in the previous test. The maximal proportion error highly influences the reached relative gap. The oscillation is caused by the simplification of the functions and occurs at the limit of precision.

In both dependence cases the speed of convergence for all parameter settings is very similar until the value of the simplification parameter begin to influence the precision.

3.7.2 Weighted flow threshold

The method of weighted flow threshold was tested on a 4x4 grid network with one OD pair. In Figure 3.10 there are the settings for the grid network. Every node proportions in the grid are set to 0.5 in order to eliminate the influence of equilibrate process. The edges have length between 1000m and 1040m. The relationship between the flow threshold and fan step was set to

$$fanStep = 2.5\ \delta_q \tag{3.37}$$

and the inflow to the network simulates the peak hour. The level of approximation (values of δ_q and δ_q^w) was set so that the computation of both methods takes similar time. First, the near-exact solution was computed. During the computation of near-exact solution, the simulation released 175 274 events and takes 3190 ms. After that, the precision of the computation was incrementally decreased, so that the run time of both methods was similar, and the average absolute error was measured. In Figure 3.11 there is the result of the test. The run time of the proposed weighted method is smaller than of the original method in all cases. In most cases, the average absolute error of the proposed method is smaller about a few orders of magnitude.

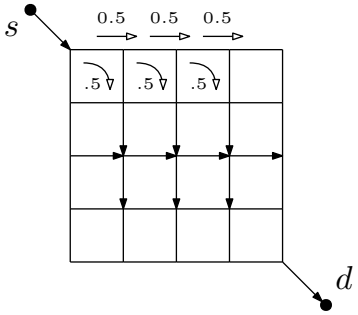


Figure 3.10: The grid network

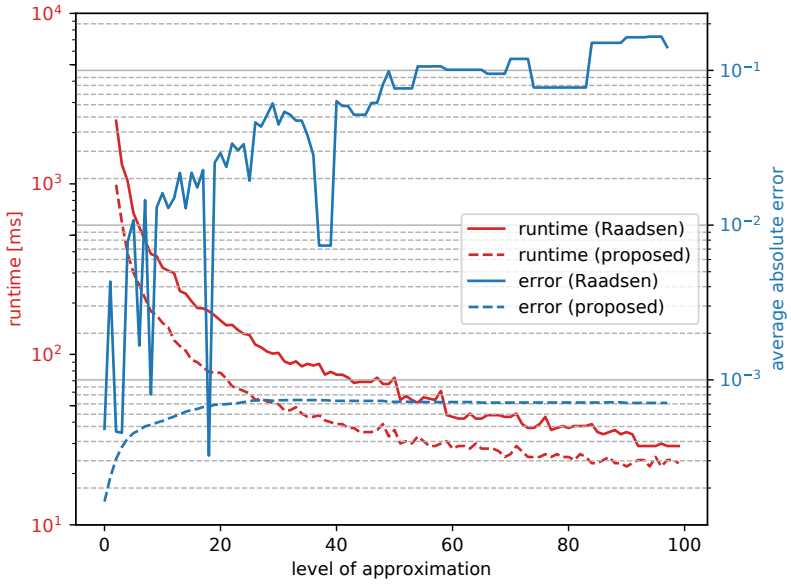


Figure 3.11: Comparison of weighted method with flow threshold method

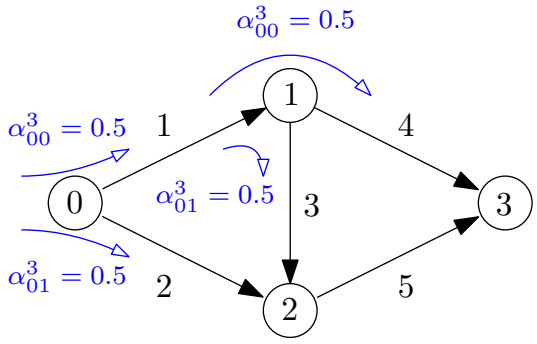
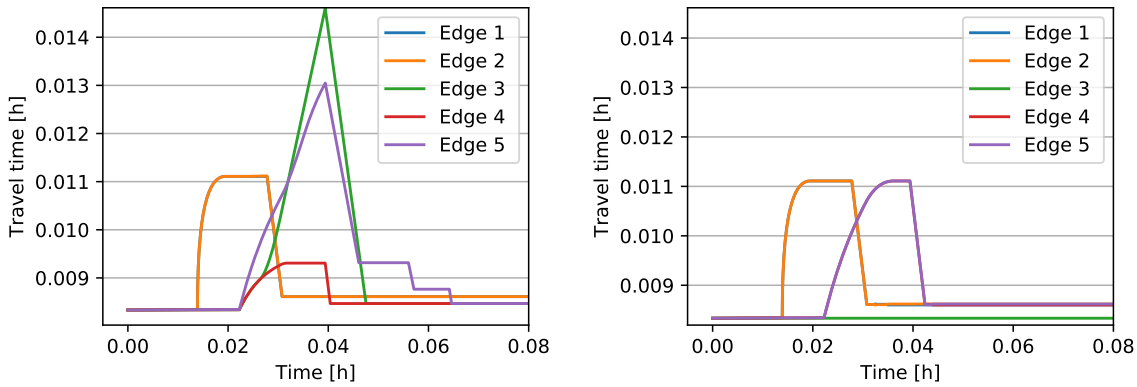


Figure 3.12: Braess network with initial settings for α^d_{ij}. The nodes and edges are labeled by numbers.

3.7.3 The Braess network

The Braess network is the classical testing use case. In Figure 3.12 there is the network with initial proportions in nodes. Every edge has length 1 km and the same FD. There is one OD pair between nodes 0 and 3 with inflow 0 veh/h until $t = 50s$, 3000 veh/h until $t = 100s$, and 500 veh/h until $t = 200s$. From these settings it follows that there are queues.



(a) TTFs for initial settings

(b) TTFs for final solution

Figure 3.13: Travel time functions for Braess network

In Figure 3.13 there are the travel time functions for the initial solution and final (equilibrated) solution. After 1000 iterations the reached relative average gap was $3.4 \cdot 10^{-4}$. This test had the aim to confirm that the solution scheme works also for networks with queues. As you can see, the flow at the edge 3 is eliminated and the travel time functions at the edge 1 and 2 (4 and 5, respectively) are very similar. The precision of the solution is perfectly sufficient for practical use.

3.7.4 Comparison of PWL/PWC approach with discrete solution

This test aims to compare the discrete solution with PWL/PWC approach on real networks. For this purpose the Anaheim network and Eastern Massachusetts by Transportation Networks for Research Core Team (2021) were used. The Anaheim network has 38 zones, 914 links and 416 nodes. The Eastern Massachusetts network has 74 zones, 258 links and 74 nodes.

For the comparison, the discrete version of the solver has been implemented. The discrete solver uses the same dynamic network loading procedure by Raadsen and Bliemer (2019) with the same setting and also uses the same implicit DUE formulation as is described in Section 3.3.1. The difference is only in the implementation of node distribution that is implemented in a classic discrete way with a defined time step. The all-to-one time-dependent shortest path problem is realized also in a discrete way for every time interval.

For both networks and for different time steps, we computed the discrete solution. The proposed PWL/PWC solution was computed using appropriate settings such that the solution provides comparable precision with the discrete case.

In Figures 3.14 and 3.15 there are the convergence curves for each setting. The solid blue color represents the proposed PWL/PWC solution and the dashed lines represent the discrete solutions. As you can see the PWL/PWC solution has the fastest convergence rate except the Anaheim with 3 min time step.

It should be noted that the relative gaps are not computed exactly in the same way because there is the influence of discretization in time-dependent shortest path search in the discrete case and in PWL TDSP case there is the influence of approximation.

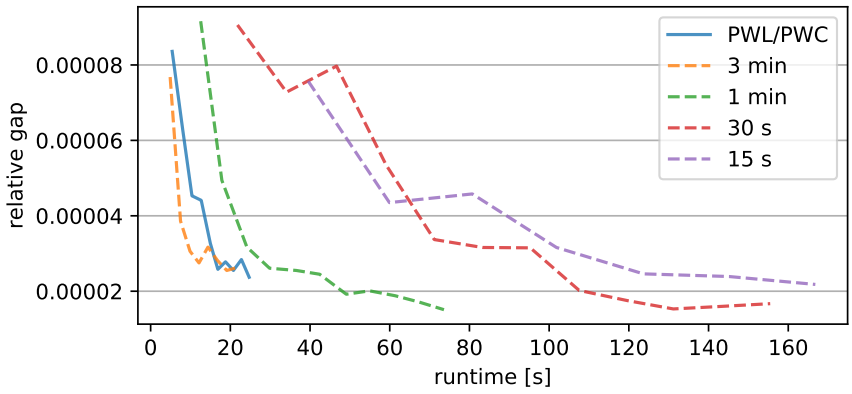


Figure 3.14: Convergence chart for Anaheim network. The time interval was set to 3 min, 1 min, 30 s and 15 s.

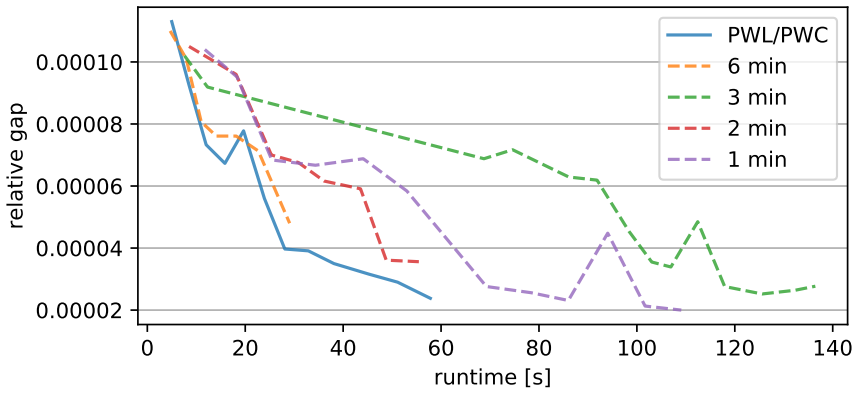


Figure 3.15: Convergence chart for Eastern Massachusetts network. The time interval was set to 6 min, 3 min, 2 min and 1 minute.

The last test tries to compare the representation of the result function α_{ij}^d. The test was performed on the Anaheim network with more accurate settings for DNL than in previous tests. In Table 3.3 there are reached gap and the average number of constant pieces of the result functions α_{ij}^d. There is also the comparison with the discrete version where the time interval/step was set to 15s, 1m, 2m, and 3m. The last column shows the size of the serialized solution on the disk. As you can see, the proposed PWL/PWC

method	χ [$\times 10^{-6}$]	# pieces	size of solution [Mb]
Discrete	8.7	480	367
PWL/PWC	5.5	19	56
Discrete	10.9	120	96
Discrete	14.4	60	50
Discrete	10.5	40	35

Table 3.3: The number of pieces/intervals that are needed for representation of result function α_{ij}^d.

method needs on average only 19 constant pieces for representation of the result.

We also tried to compare the proposed PWL/PWC solution with the classical approach that uses the LTM with the triangular diagram and the path-based formulation of DUE. We conclude that this comparison does not make any sense because the values of relative gaps and runtimes were uncomparable due to different shapes of the fundamental diagram and due to very different solution methods. The relative gaps differed about three order of magnitude.

3.8 Summary

We propose a new grid-free near continuous-time solution for the dynamic user equilibrium, that uses the approximated continuous ε-LCA-BS presented in Chapter 2. Only the simplified fanning in E-GLTM and the approximation of TTFs by PWL functions during the update phase broke the possibility of a fully continuous-time solution.

The comparison with the grid-based approach was performed and showed that the proposed solution can save computational time. Especially, these savings are significant in the case of precise discretization (short time intervals).

In the future, it would be useful to develop the marginal sequential updating of E-GLTM DNL using by Corthout et al. (2014). This should help with the convergence to equilibrium. The automatic coordination of approximation parameters and gradually increasing precision during computation is also a future research direction.

Chapter 4

Capacity based first-order node model for dynamic traffic loading

One of the submodels of dynamic network loading is the node model which can be very precise but also complex. Therefore, the choice of the model has a significant impact on the quality and performance of the whole loading procedure. This chapter tries to move the first-order macroscopic node model closer to reality by taking into account the capacity of intersection movements (internal node supply constraints) together with a relaxed First-In-First-Out rule at incoming edges such that the model stays as macroscopic as possible. We formulate the proposed model as a fixed-point problem and provide an effective algorithm that solves the problem faster than fixed-point iteration. Further, the example of the capacity model based on well-known German highway capacity manual is provided.

4.1 Introduction and State-of-the-art

As mentioned previous chapters, the analytical DTA consists of two basic models: Dynamic User Equilibrium (DUE) and Dynamic Network Loading (DNL). The DUE model defines the route choice strategy and DNL ensures vehicle propagation through the road network. The node model is a sub-model of DNL together with the link and the origin model. The link model maintains the propagation of vehicles through edges and the node model deals with propagation through the node. The common approach is to use the macroscopic first-order Lighthill-Whitham-Richards traffic model (LWR) Lighthill and Whitham (1955); Richards (1956) as the link model. In this case, the first-order node is needed because the correct propagation of flows has to be ensured. The first-order models consider only one independent variable. In our case, the independent variable is flow.

As mentioned above, the first-order node model is part of the DNL procedure which

is the most demanding part of the DTA models. There are only a few efficient state-of-the-art methods for the DNL. In particular, there are the iterative link transmission model (ILTM) by Himpe et al. (2016), the event-based link transmission model (E-LTM), and the event-based general link transmission model (E-GLTM) by Raadsen et al. (2016); Raadsen and Bliemer (2019). In the case of an inaccurate or slow node model, these high-end propagation methods are useless. It follows that the balance between computing efficiency and model precision is very important.

The model input is a demand at incoming links, a supply on outgoing links, and the information about route choice. The demand is the number of vehicles that want to enter the intersection and the supply represents the number of vehicles that can drive to the outgoing link. The model returns the number of vehicles that enter the intersection from incoming links (inflow) and the number of vehicles that leave the intersection by outgoing links (outflow). The basic requirements for the model are: the sum of vehicles entering the intersection must be equal to the number of vehicles leaving the intersection and the resulting flows must be smaller than the demand and supply (Han et al., 2018). These are expected natural constraints for flow as in the case of the maximum flow problem.

The information about the route choice has different forms depending on DUE formulation so the general formulation of the node model is needed. This formulation/definition is presented by Tampère et al. (2011) who also defined additional requirements for the first-order node models and thus a whole class of models was defined.

According to Wright et al. (2017), the methods can be divided into two epochs: pre-Tampère and post-Tampère. We mainly focus on the post-Tampère epoch. The requirements for Tampère's class of the first-order node model are fulfilled for all models mentioned below. Tampère et al. (2011) themselves published the model that, in case of congested outgoing edges, distributes the supply proportionally to the incoming capacities and presented the solution algorithm. The mechanism of supply distribution is called supply constraint interaction rule (SCIR). The possibility of the internal node supply constraints is also basically discussed. Flötteröd and Rohde (2011) builds on top of Tampère's work and proposed *incremental node model* (INM). The extension of INM is constrained INM (INMC) which incorporates the internal node supply constraints and is formulated as the fixed-point problem. Flötteröd and Rohde (2011) also pointed that the solution, in this case, does not have to be unique. The non-unique flows are discussed in detail by Corthout et al. (2012). The model by Gibb (2011) estimates the consumption of incoming capacity by vehicles that wait for crossing the intersection into the outgoing link. This approach agrees well with microsimulations and thus shifts the model close to reality. The solution method is also based on fixed-point. Smits et al. (2015) formulate the family of models as based on turn delays and publish two new models: a single server and an equal delay at the outgoing link. One year

later, Jabari (2016) shows that the maximization of the flow, as defined in (Tampère et al., 2011; Flötteröd and Rohde, 2011), is not the same as a holding-free solution. The flow maximization is a sufficient but not necessary condition. Jabari (2016) also partially resolved the problem with non-unique flows for signalized intersections using the decomposition to phases where the conflicting flows do not proceed through the intersection simultaneously. All these models mentioned above assume conservation of turning fractions (CTF) which implicates the First-In-First-Out (FIFO) rule at the incoming edges. Wright et al. (2017) pointed that this strict rule can lead to unrealistic congestion and proposed the concept of the relaxed FIFO condition which is a compromise between full FIFO conditions at incoming links and the models with no FIFO conditions. This model by Wright et al. (2017) does not deal with internal node supply constraints.

All the models mentioned above are relevant node models and can be used during network loading. For every intersection, a different model can be used. The credibility of the model, computing efficiency, and uniqueness of the flows are the basic properties of the model and should be balanced. In Table 4.1 there is a comparison of state-of-the-art models.

This chapter has the aim to incorporate the internal node supply constraints (INSC) into the model by Wright et al. (2017) and propose the effective solution algorithm. The next goal is to maintain the model as macroscopic as possible so we deal with whole edges instead of lanes. The motivation for this decision is availability of datasets that contains detailed road information as number of lanes. In this work, the internal node supply constraints are represented by a *general capacity model* which can take into account the conflicts between movement flows and/or traffic lights signal timing plans.

model	SCIR	rel. FIFO	INSC	solution	unique sol.
TAM	incoming capacities	no	no	direct	yes
INM	priorities	no	no	direct	yes
INMC	priorities	no	yes	fixed-point	no
CCE	-	no	no	fixed-point	yes
SSP	-	no	no	direct	yes
EDP	-	no	no	fixed-point	yes
WRI	priorities (zero is possible)	yes	no	direct	yes

Table 4.1: Comparison of the models: single server (SSP) and equal delay at outgoing (EDP) by Smits et al. (2015), incremental node modem (INM) and constrained INM (INMC) by Flötteröd and Rohde (2011), capacity consumption equivalence (CCE) by Gibb (2011), model by Wright et al. (2017) (WRI), model by Tampère et al. (2011) (TAM).

The remainder of this chapter is structured as follows. Section 4.2 contains notations,

definitions, problem definition, and formulation of the proposed node model. The proposed capacity model is described in Section 4.3. In Section 4.4, there is a description of three algorithms that compute the proposed model. Section 4.5 shows the examples of intersections and the solution for them.

4.2 Problem Definition and Formulation

In this section, the proposed node model is defined on the top of models by Wright et al. (2017), Tampère et al. (2011), and Flötteröd and Rohde (2011). First, definitions and notations are introduced. Then, the transformations formulas between the universal node model and the node model for the given DUE formulation are derived.

4.2.1 Notation and Requirements

Let I be the set of all incoming edges to the intersection and J the set of all outgoing edges. The set of all intersection movements is defined as $M \subseteq I \times J$. In Figure 4.1 there is an example of an intersection with two incoming and two outgoing edges.

More precisely, the model inputs are a demand S_{ij} between an incoming edge i and an outgoing edge j for all $i \in I$ and $j \in J$, a supply R_j for all outgoing edges $j \in J$. The model output is a flow q_{ij} for every intersection movement $ij \in M$. All the inputs and outputs can be understood as vectors: $S = \{S_{ij} : ij \in M\}$, $q = \{q_{ij} : ij \in M\}$, and

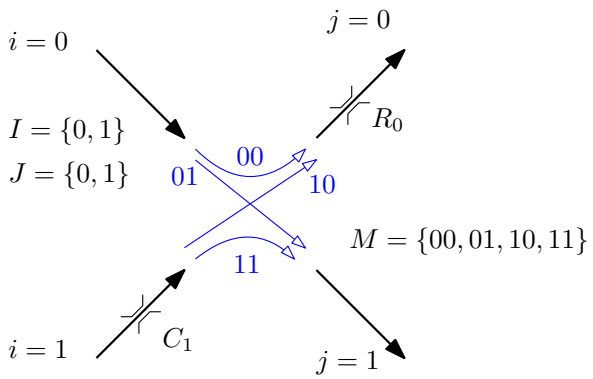


Figure 4.1: Example of intersection with two incoming and two outgoing edges

$R = \{R_j : j \in J\}$. In the whole chapter, we use the following notation:

$$q_i = \sum_{j \in J} q_{ij}; \quad S_i = \sum_{j \in J} S_{ij}$$
$$q_j = \sum_{i \in I} q_{ij}; \quad S_j = \sum_{i \in I} S_{ij}$$

The node model can be expressed as a vector function as

$$q = \theta(S, R) \tag{4.1}$$

According to Tampère et al. (2011), all first-order macroscopic node models θ should keep the following conditions:

1. General applicability for any intersection with any number of incoming and outgoing edges.
2. The maximal flow through the node assuming the flow respects the defined constraints. This is a sufficient condition for the holding-free property described by Jabari (2016).
3. Non-negativity of the flow.
4. Equality of the sum of inflow to the intersection to the sum of outflow from the intersection. This condition is called a *conservation law*.
5. Respect to demand and supply constraints. The inflow must be smaller or equal to the demand and the outflow must also be smaller or equal to the supply. Then, $q_{ij} \leq S_{ij}$ and $\sum_i q_{ij} \leq R_j$
6. Respect to the route choice. The vehicles should pass the intersection according to their route choice. The constraint is called *conservation of turning fractions* (CTF) and can be expressed as

   $$\frac{q_{ij}}{\sum_j q_{ij}} = \frac{S_{ij}}{\sum_j S_{ij}} \tag{4.2}$$

   The FIFO requirements are also included in CTF. In the model by Wright et al. (2017), this condition is not fulfilled and is replaced by a partial FIFO rule.
7. Compatibility with link flow dynamic. The model should satisfy the *invariance principle*, i.e., assuming the constant demand and supply, the flow should be invariant. This condition avoids the discontinuous changes in the flow. More information see in Lebacque (2005).

Another question is how to distribute the supply into intersection movements in case of a congested outgoing link. This mechanism is called a *supply constraint interaction rule* (SCIR).

4.2.2 Proposed Model

The aim is to develop a model which takes into account the general movement capacity model (or INSC) together with the relaxed FIFO condition by Wright et al. (2017) in order to move the model closer to reality and maintain the macroscopic properties.

The main idea of the relaxed FIFO is that for every incoming edge i, the set of *mutual restriction intervals* $\eta^i_{j'j} \in [0,1]$, which represents the degree of involvement of the FIFO rule, is defined. The interval $\eta^i_{j'j}$ means that the congestion in the outgoing edge j' affects a $|\eta^i_{j'j}|$ portion of the flow directed from the edge $i \in I$ to the outgoing edge j. It follows that the interval $[0,1]$ affects it in full (strict FIFO rule, CTF) and the interval $[0,0]$ does not affect it at all. This approach replaces condition six in the first-order macroscopic node model conditions that are defined in Section 4.2.1.

In order to satisfy the general movement capacity model providing the maximal capacity for every intersection movement, the eighth condition is added.

8. The movement flow q_{ij} must be smaller or equal to the movement capacity c_{ij}. Mathematically, $q_{ij} \leq c_{ij} \quad \forall ij \in M$.

The capacities c_{ij} can be constant or provided by some sophisticated capacity model $\mathcal{C} = (c_{ij} : ij \in M)$ which can be function of the flow q.

The model by Tampère et al. (2011) distributes the supply proportionally to capacities of incoming links. On the other hand, the Wright et al. (2017) and Flötteröd and Rohde (2011) distribute the supply proportionally to priorities p_i where $i \in I$. For our model, we use the second more general SCIR model based on priorities of incoming edges. Then, the oriented priorities of the movements are computed as:

$$p_{ij} = p_i \frac{S_{ij}}{S_i} \tag{4.3}$$

Without a loss of generality, in contrast with Wright et al. (2017), only one commodity is considered because multi-commodity issue can be resolved using simple post processing. In Section 4.2.3, this problem is discussed in detail. The proposed node model is defined as a constrained maximization problem which is

$$\max \sum_{i \in I} \sum_{j \in J} q_{ij} \tag{4.4}$$

subject to

$$q_{ij} \geq 0 \quad \forall ij \in M \tag{4.5}$$

$$\sum_{i \in I} q_{ij} \leq R_j \quad \forall j \in J \tag{4.6}$$

$$q_{ij} \leq c_{ij} \quad \forall ij \in M \tag{4.7}$$

$$q_{ij} \geq \frac{p_{ij}}{\sum_{i' \in I} p_{i'j}} R_j, \quad \forall i \in \{i : W_i \neq \emptyset\},\ \forall j \in W_i \tag{4.8}$$

$$q_{ij} \leq S_{ij} - \mathcal{A}\left(\bigcup_{j' \in \mathcal{W}_i \setminus \{j\}} \mathcal{Q}^i_{j'j} \right) \quad \forall ij \in M \tag{4.9}$$

where rectangle

$$\mathcal{Q}^i_{j'j} = \eta^i_{j'j} \times \left[\frac{q_{ij'}}{S_{ij'}} S_{ij}, S_{ij} \right] \tag{4.10}$$

and according to Wright et al. (2017), $\mathcal{W}_i = \{j : R_j > S_{ij} > 0, \exists i' \neq i : p_{i'j} q_{ij} \geq p_{ij} q_{i'j}\}$ is the set of restricting output edges for the incoming edge $i \in I$, $\eta^i_{j'j}$ is the mutual restriction interval, $\mathcal{A}$ represents the area of two-dimensional object, and finally '$\times$' makes rectangle defining by two intervals.

The inequalities (4.5)-(4.6) correspond with requirements 3 and 5 from Section 4.2.1. The constraint (4.7) maintains that the resulting flow is not higher than movement capacities. The SCIR (priority constraints), where the supply is distributed proportionally to the intersection movement oriented priorities, p_{ij} is expressed in (4.8). Finally, the (4.9) ensures the relaxed FIFO mechanism. Here, the area of rectangles union is computed. The (4.8) and (4.9) are taken from Wright et al. (2017). For a detailed description of the equations see the original paper.

In the case of an unsignalized intersection (intersection without any traffic lights and any other controls) and partially also for a signalized intersection, the capacity of movement c_{ij} is generally dependent on the flow at other movements. The movement capacities c_{ij} are the result of a capacity model that can be generally written as

$$c = \mathcal{C}(q) \tag{4.11}$$

If the capacity model is included into the node model, the node model can be written as

$$q = \theta(S, R, q) = \theta(q) \tag{4.12}$$

As can be seen, the resulting flow q is also an argument of the model. This formulation corresponds with the *fixed point problem* (FPP). This fact complicates the solution. The same formulation is used by Flötteröd and Rohde (2011).

4.2.3 Models for DUE Formulations

As mentioned above, the inputs for the node model are: the demand between the incoming and the outgoing edges S_{ij} and the supply at the outgoing edges R_j that can be directly obtained from the link model.

However, the demand S_{ij} must be computed based on the dynamic user equilibrium (DUE) formulation. There are two basic DUE formulations. One is a classical formulation with an explicit path enumeration (path-based) and the other is with an implicit path formulation (link-based Ran et al. (1996), destination-based Gentile (2016), intersection-movement-based Long et al. (2013)). In the next two subsections, the computation of demand for different formulations is presented.

Implicit Formulation

All the implicit DUE formulations are similar. The key variable driving the route choice is node distribution matrix $A^d = \{\alpha_{ij}^d : i \in I, j \in J\}$. The distribution α_{ij}^d gives the proportion of flows which direct to the destination d through the outgoing edge j. It follows that $\sum_j \alpha_{ij} = 1$.

Next important variable is the proportion of flow m_i^d at the target node of incoming edge i that directs to the destination d. This variable is called a *mixture*. Then one cell of the total node distribution matrix $A = \{\alpha_{ij} : i \in I, j \in J\}$ can be written as

$$\alpha_{ij} = \sum_d m_i^d \alpha_{ij}^d \tag{4.13}$$

The α_{ij} gives the proportion of flows on the incoming edge i which direct to outgoing edge j. Now the demand can be simply calculated as

$$S_{ij} = S_i \alpha_{ij} \tag{4.14}$$

Now the flow q_{ij} can be computed using some node model θ.

The computation of mixture at the outgoing edges is also a task for the node model. Based on variables defined above, the mixture can be written as

$$m_j^d = \frac{\sum_i \frac{m_i^d \alpha_{ij}^d}{\alpha_{ij}} q_{ij}}{q_j} \tag{4.15}$$

where q_{ij} is resulting flow and $q_j = \sum_{i \in I} q_{ij}$.

The destination d can be understood as a commodity. In this case, this mechanism solves the multi-commodity issue.

Path-based Formulation

The key variable that tracks the mixture is a proportion of flow μ_i^p at the incoming edge i associated with the path p, see Han et al. (2018). All flows which direct to the same outgoing edge from the given incoming edge must be summed. Then

$$S_{ij} = S_i \sum_{p \ni i,j} \mu_i^p \tag{4.16}$$

After the node model is performed, the path proportion at the outgoing edges should be calculated based on the node model results as

$$\mu_j^p = \frac{S_i \mu_i^p \frac{q_{ij}}{S_{ij}}}{q_j} \tag{4.17}$$

This proportion μ_j^p will be used by the link model in the next time intervals. The described operations can be understood as a kind of transformation between the DUE formulation and the universal node model that is defined in Section 4.2.1.

4.3 Capacity Model

Let the function $\mathcal{C} : \mathbb{R}^n \to \mathbb{R}^n$ be the capacity model that represents the capacity of the movements. Generally, the capacity of movement from the incoming edge i to the outgoing edge j can be dependent on all other movements flow. So

$$c_{ij} = \mathcal{C}_{ij}(q) \tag{4.18}$$

This dependence can be caused by movement conflicts. In the next two sections, the capacity model for the signalized and the unsignalized intersection is described.

4.3.1 Unsignalized

The rules on the unsignalized intersection are defined by traffic signs "Stop", "Give way", or "Priority road" and priority of the right. The major and minor roads can be defined in this type of junction based on the traffic signs.

For example, the T-shaped intersection with two incoming edges and one outgoing edge is considered. There is one major movement from the edge $i = 0$ to $j = 0$ and one minor movement from the edge $i = 1$ to the edge $j = 0$. The vehicles incoming from the minor edge $i = 1$ must give way to cars that cross the intersection by the major road. It follows that the capacity of the minor movement is dependent on the flow at the major movement.

Generally, according to (4.18) the capacity c_{ij} can depend on all superior movements. The question is how to resolve the dependencies. The dependence of the intersection

movement ij on kl can be understood as an edge of the graph where the movements are the nodes of the graph. Formally, the set of edges is defined as

$$E = \{kl \to ij : ij \text{ depends on } kl \quad \forall ij, kl \in M\} \tag{4.19}$$

then the graph can be defined as $G = (M, E)$. The nodes (movements) of the graph must be ordered so that each node depends only on the nodes in the order before it. The following lemma says under what conditions the movements can be sorted.

Lemma 3. *Let G is an acyclic graph then the capacity of movements can be directly resolved using topological ordering.*

The lemma says that if the graph G contains a cycle, the movements cannot be sorted. This situation occurs for example if movement ij depends on kl and kl depends on ij. These dependencies create the cycle with length two. In reality, the graph has to be acyclic because the traffic rules define who has priority of driving.

Example of a Capacity Model

This section introduces the capacity model that is based on Czech technical norms (Technické podmínky Ministerstva dopravy) (TP188, 2018) and German Highway Capacity Manual (Forschungsgesellschaft für Strassen und Verkehrswesen, 2001).

First, the superior movement flow q_{ij}^H that represents the sum of conflicts flow, is defined as

$$q_{ij}^H = \sum_{kl \in M} \beta_{ij}^{kl} q_{kl} \tag{4.20}$$

where $\beta_{ij}^{kl} \in [0, 1]$ is a level of dependence $ij \in M$ on $kl \in M$. If $\beta_{ij}^{kl} = 0$ then there is no dependence. Based on the level of dependence, the set of edges E of the graph G can be written as

$$E = \{kl \to ij : \beta_{ij}^{kl} > 0 \quad \forall ij, kl \in M\} \tag{4.21}$$

Using the superior movement flow q_{ij}^H, the capacity of the movement from the incoming edge i to the outgoing edge j is defined as follows (TP188, 2018; Forschungsgesellschaft für Strassen und Verkehrswesen, 2001)

$$c_{ij} = \mathcal{C}_{ij}(q_{ij}^H) = \frac{3600}{t_f} e^{-\frac{q_{ij}^H}{3600}\left(t_g - \frac{t_f}{2}\right)} \tag{4.22}$$

where t_g [s] is the critical time interval, t_f [s] is the next time interval. The parameters t_g and t_f should be set according to Table 4.2. If the capacity is smaller than the initial flow q_{ij} then the flow between i and j is updated as

$$q_{ij} = \min(c_{ij}, q_{ij}) \tag{4.23}$$

intersection movement	t_g [s]	t_f [s]	
		"Give way"	"Stop"
left turn from the main road	$t_g = 3.4 + 0.021v$	2.6	2.6
right turn from minor road	$t_g = 2.8 + 0.038v$	3.1	3.7
straight passage from the minor road	$t_g = 4.4 + 0.036v$	3.3	3.9
left turn from the minor road	$t_g = 5.2 + 0.022v$	3.5	4.1

Table 4.2: Critical time interval and next time interval values by TP188 (2018). The variable v, $v \in [30, 90]$, is speed on main road in km/h.

The solution algorithm is straightforward. First, the topological ordering of movements is performed and then the capacities are gradually calculated.

The values t_g and t_f in Table 4.2 are calibrated for rules and driver behavior in the Czech Republic. To be used in different countries, the recalibration should be performed.

4.3.2 Signalized

The signalized intersection can be decomposed into phases (cycles). According to Jabari (2016), within one phase, the flow can be solved separately, and thereby the solution is simplified and with additional assumptions leads to a unique solution. As is mentioned above, we aim to maintain the model as macroscopic as possible. We are focusing on the continuous modeling of a signalized intersection.

The capacity of the movement depends on the time length of the phase and on conflicts between movements within phase. In the case of no conflict, the capacity can be computed by TP188 (2018) as

$$c_{ij} = q_{ij}^S \frac{z}{t_c} \tag{4.24}$$

where z represents the length of the green time window, t_c is the time length of cycle (phase), and q_{ij}^S is the saturated flow for movement $ij \in M$. In case of conflict, for example, a left turn is often in conflict with a major movement stream[1], the capacity is further reduced by mechanisms from the previous section.

The detailed description of continuous modeling of a signalized intersection is out of the scope of this work. We refer to Yahyamozdarani et al. (2019) for further reading.

[1] We assume that cars drive on the right side of the road.

4.4 Solution Algorithms

This section contains a description how to solve the first-order macroscopic node model that is proposed in Section 3.3.1 and represented by expressions (4.4)-(4.9). We denote the proposed model as constrained Wright's model (WRIC). We use the idea that was proposed by Flötteröd and Rohde (2011). First, the demand S is bounded by *demand constraint function* using movement capacities c_{ij} and then a standard algorithm which does not take into account the INSC is used. In our case the algorithm is the Wright's MIMO algorithm (WRI).

If the CTF (full FIFO) is assumed, the bounded demand is determined as

$$\widehat{S}_{ij} = S_{ij} \min\left(1, \min_{j' \in J} \frac{c_{ij'}}{S_{ij'}}\right) \tag{4.25}$$

The most constrained intersection movement affects all other movements that origins at the same incoming edge. The demand on other movements is bounded by the same most restricting reduction coefficient.

In the case of relaxed FIFO, the demand constraint function is not so straightforward. From the model definition, we know that the inequality (4.9) drives the relaxed FIFO mechanism. The demand on intersection movement ij must be bounded by capacity c_{ij} and possibly is partially affected (bounded) by reductions on other movements leading from the same incoming edge i. The level of reduction depends on mutual restriction intervals. So we can write:

$$\widehat{S}_{ij} \leq c_{ij} \tag{4.26}$$

$$q_{ij} \leq S_{ij} - \mathcal{A}\left(\bigcup_{j' \in W_i \setminus \{j\}} \mathcal{Q}^i_{j'j}(q_{ij'})\right) \leq S_{ij} - \mathcal{A}\left(\bigcup_{j' \in W_i \setminus \{j\}} \mathcal{Q}^i_{j'j}(c_{ij'})\right) \geq \widehat{S}_{ij} \tag{4.27}$$

where

$$W_i = \{j : c_{ij} < S_{ij},\ j \in J\} \tag{4.28}$$

$$\mathcal{Q}^i_{j'j}(q_{ij'}) = \eta^i_{j'j} \times \left[\frac{q_{ij'}}{S_{ij'}} S_{ij}, S_{ij}\right] \tag{4.29}$$

The most restricted constrain is selected so we get the bounded demand which is

$$\widehat{S}_{ij} = \min\left(c_{ij}, S_{ij} - \mathcal{A}\left(\bigcup_{j' \in W_i \setminus \{j\}} \mathcal{Q}^i_{j'j}(c_{ij'})\right)\right) \tag{4.30}$$

In case of no-FIFO ($\forall i, j, j' : |\eta^i_{j'j}| = 0$), the (4.30) is reduced to $\widehat{S}_{ij} = \min(c_{ij}, S_{ij})$ and in case of full-FIFO it is reduced to (4.25). Together with (4.12) and (4.11), the fixed-point problem is

$$q = \theta(\widehat{S}_{ij}(S, \mathcal{C}(q))) \tag{4.31}$$

According to Flötteröd and Rohde (2011), the fixed-point problem (4.31) has at least one solution and according to Corthout et al. (2012), there can be multiple solutions. Wright's algorithm provides the maximal solution and from (4.26), (4.27), and (4.30), it directly follows that the bounded demand $\widehat{S}$ is also maximal.

This formulation is identical with equation (27) by Flötteröd and Rohde (2011) who also proposed two algorithms for how the problem solve: exact and approximate solution procedure. The exact solution algorithm assumes that the incoming edges can be ordered such that the edge i is independent of edges $i+1, \ldots, |I|$. This assumption can not be fulfilled for various intersections including also very simple ones. The second algorithm provides an approximate solution that is computed using the linearization between two working points. We incorporated this approximate approach to a fixed-point iteration procedure and thus we can obtain the exact solution. In the following text, this method is noticed as *Flotterod's algorithm* (FL).

The simplest way is to use the *fixed-point iteration* (FP) or some acceleration method directly to equation (4.31). The root-finding algorithms can be also used. We provide solution algorithm base on *Steffensen's algorithm* (ST) (Algorithm 6) by Steffensen (1933). The idea is to apply the Aitken's Δ^2 method (Aitken, 1927) to every movement flow separately. It follows that the algorithm do not take into account cross movement dependencies.

Algorithm 6: Steffensen's algorithm based solution for proposed model

```
1 q = S                                          first approximation
2 δ = ∞                                          set current gap infinity
3 while δ > ϵ do
4     q^a = θ(Ŝ_ij(S, C(q)))                     run Wright's algorithm for the first time
5     q^b = θ(Ŝ_ij(S, C(q^a)))                   run Wright's algorithm for the second time
6     for ij ∈ M do
7         q_ij^{k+1} = (q_ij^a − q_ij^k)^2 / (q_ij^b − 2q_ij^b + q_ij^k)   Aitken's delta squared method
8     δ = |q^k − q^{k+1}|                        compute gap
9     k = k + 1                                  increase iteration counter
```

First, the iteration counter k is set to zero and the gap of the solution δ is set to infinity. As the first approximation (the starting point) of the flow q, the demand S can be chosen. The algorithm is running until the gap δ is not smaller than the maximum allowed gap ϵ. During the DNL computation, the node model is called many times so the implementation of the algorithms must be as efficient as possible. However, we can estimate the first approximation of the flow as the previous result and thus reduce the number of iteration.

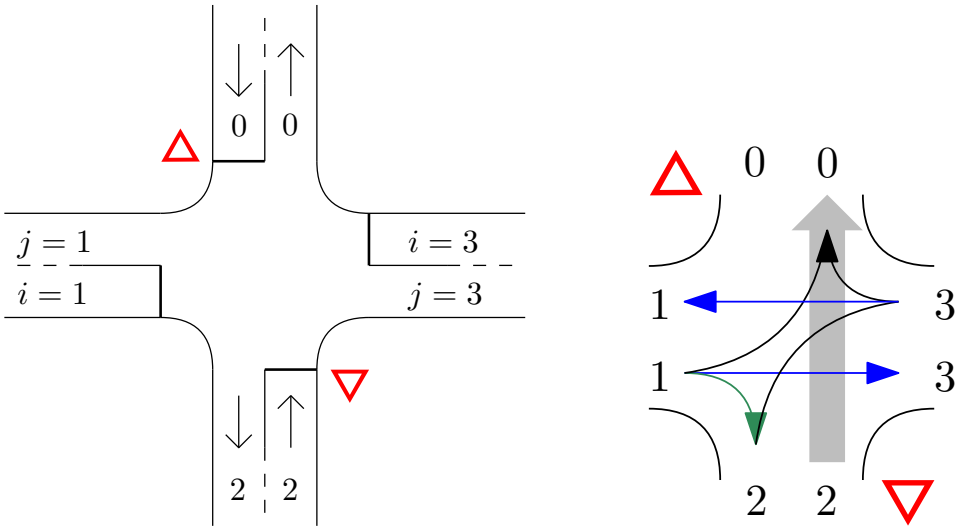
4.5 Examples and Tests

All the algorithms were implemented using Scala programming language and the codes were included into a library that focuses on computing dynamic traffic assignment. The tests were performed on a computer with a quad-core processor (Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz) and with 16GB memory.

4.5.1 Performance of the Algorithms

The aim of this test is to compare the performance of three algorithms: fixed-point iteration (FP), iterative version of Flotterod's algorithm (FL) and modified Steffensen's algorithm (ST).

We assume a classic X-shaped intersection. In Figure 4.2a there is a visualization of the intersection with edge identifications. In this test, we assume the full FIFO rule, this means that $\forall i \in I, j \in J, j' \in J : \eta^i_{j'j} = [0, 1]$. Further, the dependencies between movements must be determined. For this purpose, the methodology by TP188 (2018) and Forschungsgesellschaft für Strassen und Verkehrswesen (2001) was used. In Table 4.4 there are the non-zero dependence coefficients β^{kl}_{ij}.



(a) main and minor road (b) movements conflicts

Figure 4.2: Four incoming four outgoing intersection

For example, in Figure 4.2b, the movement 20 is in conflict with the straight movements 13 and 31 (blue color) and turn movements 32, 30, and 10 (black color). The turn 13 (green color) also influences the capacity of 20 but only halfway. In Figure 4.3 you can see the whole acyclic dependency graph. In case of the assumed intersection,

there are four levels of movements. First, the movements at the top are resolved and then the capacities for the movements 21 and 03 are computed.

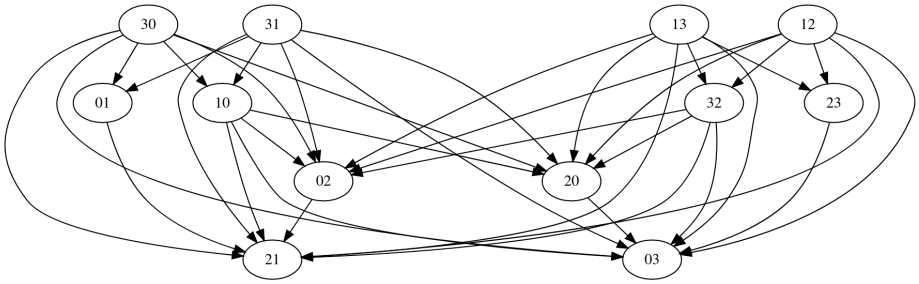


Figure 4.3: Dependency graph

We simulate a fully congested intersection, because, in this case, the algorithms have the worst performance. We set $R = (1000, 2000, 1000, 2000)$ and S is big enough so that every incoming edge is congested. Due to the invariance principle, the changes in demand do not affect the solution. For testing purposes, $\delta = \sum_{ij \in M} |q_{ij}^k - q_{ij}^{k+1}|$ and

	$\epsilon = 10^{-10}$		$\epsilon = 1.0$	
method	# iter.	δ $[\times 10^{-14}]$	# iter.	δ
FP	1366	9 984	338	0.98
FL	17	4 289	6	0.10
ST	7	9	5	0.54

Table 4.3: The number of iterations and δ reached for two settings of maximal error ϵ

we set the maximal error ϵ to 1 and 10^{-10} veh/h. In Table 4.3 you can see that FP method needs in comparison with the other two methods huge number of iterations. It should be pointed out that the FL as well as ST algorithms need two evaluations of WRI algorithm per one iteration. Despite that they are much faster. We received similar results also for other intersections.

Raadsen and Bliemer (2019), in their continuous loading procedure, recommend setting the flow threshold between 2 and 5 veh/h. It follows that the maximal error $\epsilon = 1$ veh/h is enough for this purpose.

4.5.2 Capacity-Constrained Intersection with Relaxed FIFO

The second test is focused on accuracy comparison of existing node macro models in case that the intersection has multilane incoming edges and there are movement flow conflicts. We assume the same intersection as in Section 4.5.1 with a few changes. The incoming edges 1 and 3 are considered as two lanes and there are the traffic lights with

ij	kl	β_{ij}^{kl}	ij	kl	β_{ij}^{kl}	ij	kl	β_{ij}^{kl}	ij	kl	β_{ij}^{kl}	ij	kl	β_{ij}^{kl}	ij	kl	β_{ij}^{kl}
10	31	1.0	01	31	1.0	20	10	1.0	02	10	1.0	21	10	1.0	03	13	1.0
10	30	1.0	01	30	0.5	20	32	1.0	02	32	1.0	21	32	1.0	03	12	0.5
32	13	1.0	20	13	1.0	02	31	1.0	21	13	1.0	21	01	1.0	03	10	1.0
32	12	1.0	20	12	0.5	02	30	0.5	21	12	0.5	21	02	1.0	03	32	1.0
23	13	1.0	20	31	1.0	02	13	1.0	21	31	1.0	03	31	1.0	03	23	1.0
23	12	0.5	20	30	1.0	02	12	1.0	21	30	0.5	03	30	0.5	03	20	1.0

Table 4.4: Coefficients for X intersection

signal timing plan that contains two phases. During the first phase, all intersection movements originating at incoming edges 1 and 3 are open. The second phase contains the rest of the movements originating at edges 0 and 2. Each phase takes 45 seconds (35s green and 10s yellow). These phases are independent so we can model every phase separately and then merge them into one result. In the following text, we are interested in the first phase only. In Figure 4.4 there are open movements during the first phase and the used demand.

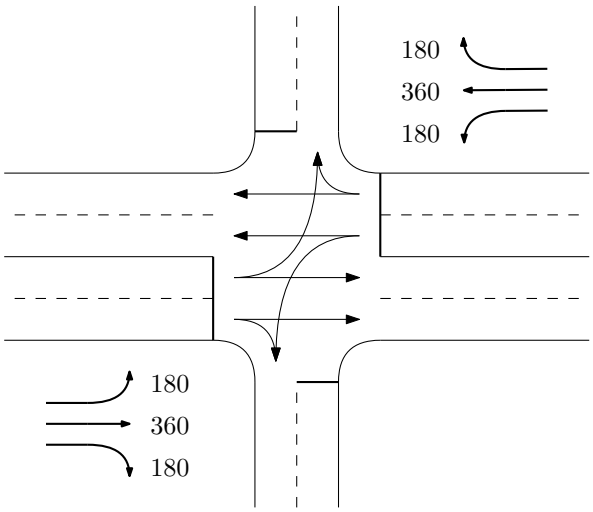


Figure 4.4: First phase with demand [veh/h]

The capacity of the movements are determined by the capacity model from Section 4.3 with the fact that the input flow to the model was multiplied by coefficient $\kappa = 2.5$. This is justified because the time length of red lights concentrates the number of vehicles and thus increases the flow. The result capacities are also divided by this coefficient κ. Mathematically, $c = \frac{\mathcal{C}(\kappa q)}{\kappa}$.

The mutual restriction intervals are set as follows: $\eta_{03}^1 = [0, 0.35]$, $\eta_{23}^1 = [0.65, 1.0]$, $\eta_{02}^1 = \eta_{20}^1 = [0, 0]$, $\eta_{01}^3 = [0.65, 1.0]$, $\eta_{21}^3 = [0, 0.35]$, and $\eta_{02}^3 = \eta_{20}^3 = [0, 0]$. This means

that the movements 12 and 10 do not affect each other and affect the movement 13 partially. Symmetrically, the same applies to incoming edge 3.

To determine near-real result flow, we use the microscopic traffic simulation software Eclipse SUMO - Simulation of Urban Mobility in version 1.9.2. (Lopez et al., 2018). We compare the result flows provide by INMC, WRI and WRIC (proposed model). In Table 4.5, you can see the differences for each movement compared to the flow computed by SUMO. The Δq represent the difference and $\delta = \sum_{ij \in M} |\Delta q_{ij}|$.

	movement flow [veh/h]						movement flow error [veh/h]						δ
	q_{10}	q_{12}	q_{13}	q_{30}	q_{31}	q_{32}	Δq_{10}	Δq_{12}	Δq_{13}	Δq_{30}	Δq_{31}	Δq_{32}	
SUMO	99	180	329	180	302	100	-	-	-	-	-	-	0
INMC	132	132	265	132	265	132	-33	48	64	48	37	-32	262
WRI	180	180	360	180	360	180	-81	0	-31	0	-58	-80	250
WRIC	103	180	306	180	306	103	-4	0	23	0	-4	-3	34

Table 4.5: Model comparisons. INMC - constrained incremental node model by Flötteröd and Rohde (2011), WRI - model with relaxed FIFO by Wright et al. (2017), WRIC - proposed model (Wright's model with INSC)

The model by Wright et al. (2017) does not take into account the INSC and thus the flow is not constrained and is equal to demand. In contrast, the INMC model follows strict FIFO, and thus the flow is too constrained. The proposed model (WRIC) is in the middle between INMC and WRI. WRIC also has the smallest error compared to SUMO flows. It should be pointed that this decomposition to the phases does not fulfill the requirements proposed by Jabari (2016) so the solution does not have to be unique.

The proposed WRIC node model shifts the macro models closer to reality. However, the model requires more computational time, because the fixed-point problem must be solved. It follows that this model should be used for important intersections. For less frequented intersections the existing models are precise enough.

4.6 Summary

We introduced the first-order macroscopic node model that takes into account the internal node supply constraints and the relaxed FIFO rule at incoming edges. This proposed model extends model by Wright et al. (2017) by the possibility of the general capacity. An example of the capacity model that takes into account the conflict between intersection movements is also provided. This example capacity model is based on TP188 (2018) and Forschungsgesellschaft für Strassen und Verkehrswesen (2001). The comparison with flow provided by micro-simulation software SUMO shows that the proposed WRIC model fits well to this micro-simulation result.

The fixed point problem was used as a mathematical formulation. We compared three algorithms for solving this problem and these algorithms are fixed point iteration (FP), iterative version of algorithm by Flötteröd and Rohde (2011) (FC) and the algorithm based on Steffenson's method (ST). Our test shows that the ST method provides better performance than the other two algorithms.

The disadvantage of the algorithms reflecting INSC is that they must be solved iteratively and thus performance is poor. This issue can be solved by the strategy where the previous result flow in simulation is used as the first approximation of the solution. However, there are still problems with the uniqueness of solution.

Chapter 5

Origin-destination matrix estimation using bush-based user equilibrium algorithms

This chapter deals with origin-destination matrix estimation using traffic counts where the traffic assignment subproblem is solved using modern bush-based algorithms that are very powerful. Compared to the previous chapters, the static user equilibrium is assumed here. The proposed algorithm tries to remove the disadvantages of the path-based algorithms for the origin-destination matrix estimation. For this purpose, the presented algorithm uses the well-known advantages of bushes (the origin-rooted acyclic subgraph) so that the algorithm does not need to enumerate the paths between origins and destinations.

5.1 Introduction

The origin-destination matrix (ODM) estimation (also called calibration) is a very important step in transportation modeling. Unfortunately, it is also a very time-consuming step. There is a lot of methods how to estimate the matrix , e.g., using speed data, traffic counts, and partial path data (license plates). In this chapter, we deal with ODM estimation using traffic counts. The input is observed traffic flow on selected links and an initial ODM. The output is the calibrated matrix.

After the year 2000, the bush-based algorithms for solving user equilibrium (UE) came. These algorithms provide the equilibrated flow on acyclic subgraphs called bushes so these algorithms compute the UE implicitly without path enumeration and they seem the most powerful class of algorithms for solving UE (Perederieieva et al., 2015).

In the classic formulations of ODM estimation, the flows on paths between origin-destination pairs are needed to compute the search direction during the matrix esti-

mation. It is well-known that the solution of UE in the path space is not unique and the maximum entropy UE (MEUE) should be computed (Xie and Nie, 2019). The best-known method for solving MEUE is also bush-based and computes the solution without the path enumeration.

Our proposed method builds on bush-based methods and estimates the ODM also without path-enumeration and thus saves the memory and computational time.

In Section 5.2 there are definitions of the problem and all subproblems with a literature survey. Section 5.3 introduces the proposed method for the determination of the assignment matrix. The numerical tests are described in Section 5.4.

5.2 Problem definition and state-of-the-art

The task is to estimate (calibrate) the origin-destination matrix using the traffic counts. The inputs are non-calibrated (target or initial) ODM and traffic counts on selected edges in the road network.

Let $G = (V, E)$ be a directed graph that represents the road network, where V is a set of nodes and E is a set of edges. The set of zones $Z \subset V$ contains all nodes where the vehicles enter/leave the road network. The $g = \{g_{ij} : ij \in W\}$ is the origin-destination matrix (ODM) containing the number of trips between all zones, where W is a set of all origin-destination (OD) pairs. The g_{ij} is the number of trips between the origin zone $i \in Z$ and the destination zone $j \in Z$. The $Q = \{g : g_{ij} \geq 0\}$ is a set of all feasible ODMs.

5.2.1 Origin-destination matrix estimation

Let $\widehat{v} = (\widehat{v_a} : a \in \widehat{V})$ be a vector of observed traffic flows where $\widehat{E} \subset E$ is a set of edges where the traffic flow was measured. According to Lundgren and Peterson (2008) the origin-destination matrix estimation problem is defined as bi-level optimization problem as

$$\min_g F(g) = \gamma_1 F_1(g, \widehat{g}) + \gamma_2 F_2(v(g), \widehat{v}) \tag{5.1}$$

where $\widehat{g}$ is the initial (target) ODM and $v(g)$ is a function that assigns the ODM to the road network. This function provides the static user equilibrium (see next section). The functions F_1 and F_2 return the distance between vectors and can be defined in various way.

The formulation of the objective function F based on entropy maximization was published by Van Zuylen and Willumsen (1980); van Zuylen and Branston (1982). The maximum likelihood approach was presented by Spiess (1987). Cascetta (1984) used the objective functions based on generalized least squares. The Bayesian inference approach by Maher (1983) and Dey and Fricker (1994) provides a method for combining of two

sources of information. Solution based on classic least squares was published by Spiess (1990), Lundgren and Peterson (2008), and Rostami Nasab and Shafahi (2020).

For our purpose, we simply choose $\gamma_1 = 0$ and

$$F = F_2 = \frac{1}{2}\sum_{a \in \widehat{E}} (v_a - \widehat{v_a})^2 \tag{5.2}$$

as in Spiess (1990). There is a lot of methods how to solve this bi-level problem. The general approach is to use the steepest decedent with a long step (Lundgren and Peterson, 2008; Spiess, 1990). A general solution strategy consists from three steps:

- Compute search direction. The simplest method is to take the negative value of the gradient.
- Determine the size of a step so that the objective function is minimized.
- Update the ODM using the search direction and the step size.

The most difficult task is to compute the gradient of F_2, namely, the value of $\frac{\partial v_a}{\partial g_{ij}}$ (Lundgren and Peterson, 2008). For computation of these values there is several heuristic methods (see Lundgren and Peterson (2008)). A common feature of these methods is that they need the *assignment matrix*. The assignment matrix expresses the relationship between the edge flow v_a and OD flow g_{ij}.

In this chapter we present an effective way how to compute the assignment matrix implicitly without path enumeration. For this purpose, the origin-rooted flow provided by bush-based algorithms, are used.

5.2.2 User equilibrium

Let the cost c_a of the edge $a \in E$ is dependent on the traffic flow v_a

$$c_a = c_a(v_a) \tag{5.3}$$

It is assumed that the cost function $c_a(v_a)$ is monotonically increasing and convex. The user's route choice is dependent on travel costs. It follows that equilibrium must be found. The user equilibrium is the state where every used path from the source zone i to the destination zone j has an equal (minimal) cost. The problem of searching user equilibrium is called Traffic Assignment Problem (TAP) and can be defined as Variational Inequality problem (VI) (Dafermos, 1980; Smith, 1979). The optimal solution v_p^* must satisfy

$$\sum_{p \in P} c_p(v_p^*)(v_p - v_p^*) \geq 0, \quad \forall u \in \Lambda \tag{5.4}$$

where v_p is the flow on the path p, P is the set of all used paths in the road network, c_p is the cost of path p, $u = (v_p : p \in P)$ is a vector of all path flows, and Λ is the set of all feasible solutions in the space of paths

$$\Lambda = \left\{ u > 0 : \sum_{p \in P_{ij}} v_p = g_{ij} \quad \forall ij \in W \right\} \tag{5.5}$$

where $P_{ij} \subset P$ is a set of used paths for OD pair $ij \in W$. The relationship between the solution in the path space and the edge space is

$$v_a = \sum_{p \in P} \delta_{ap} v_p \tag{5.6}$$

where $\delta_{ap} \in \{0, 1\}$ is equal to one if the edge a lies on the path p otherwise the δ_{ap} is zero. It should be noted that there is only one solution in edge space. In contrast with this the solution in path space is not unique (Xie and Nie, 2019).

According to Perederieieva et al. (2015), there are three basic groups of algorithms for solving TAP:

- *Link-based* algorithms compute the solution in the edge space. They have low memory requirements but very poor convergence rate. A classic representative of this group is Frank-Wolfe algorithm (FW). e.g., (LeBlanc et al., 1985).
- *Path-based* algorithms search the solution in the path space. They are much faster than the link-based algorithms but generally need a lot of memory for the path enumeration. e.g, (Xie et al., 2018; Babazadeh et al., 2020).
- *Bush-based* algorithms decompose the problem to acyclic sub-graphs called *bushes*. These algorithms are fast and do not need to enumerate the paths. e.g., (Bar-Gera, 2002; Dial, 2006; Gentile, 2014; Bar-Gera, 2010).

As was shown by Perederieieva et al. (2015), the bushed-based algorithms generally provide good performance.

5.2.3 Maximum Entropy User Equilibrium

As mentioned above, the assignment matrix and the flow on paths are necessary for ODM estimation, but the solution in path space is not unique. We must choose the solution with the most likely realization that is generally understood in terms of maximum entropy. According to Xie and Nie (2019), the maximum entropy user equilibrium (MEUE) is defined as

$$\max_{v_p} = - \sum_{ij \in W} \sum_{p \in P_{ij}} v_p \log \left(\frac{v_p}{g_{ij}} \right) \tag{5.7}$$

subject to

$$\sum_{p \in P_{ij}} v_p = g_{ij}, \quad \forall ij \in W \tag{5.8}$$

$$\sum_{p \in P} \delta_{ap} v_p = v_a, \quad \forall a \in E \tag{5.9}$$

$$v_p \geq 0, \quad \forall p \in P \tag{5.10}$$

In literature, two methods can solve the MEUE problem for real size networks. The first method by Xie and Xie (2016) uses the *paired alternative segments* that are provided by TAPAS algorithm (Bar-Gera, 2010). The second method by Xie and Nie (2019) uses the flow on bushes and maximizes entropy on them so the method does not enumerate the paths. As was shown in Xie and Nie (2019), the second method provides better results.

5.3 Proposed solution

In this section, the implicit method for computing the assignment matrix is introduced. For this purpose, the flow on bushes is used. This bush-based flow has to fulfill the user equilibrium and maximizes the entropy. In this chapter, the B-algorithm by Dial (2006) is used for providing the equilibrated bash-based flow. For entropy maximization, the algorithm by Xie and Nie (2019) was implemented.

For purpose of this chapter, we build on the estimation approaches according to Spiess (1990) because it is simple and effective for large networks (Lundgren and Peterson, 2008). However, the idea of assignment matrix computation can be applied to various estimation methods.

According to Spiess (1990) the gradient is computed as

$$\frac{\partial F(g)}{\partial g_{ij}} = \sum_{k \in P_{ij}} p_k \sum_{a \in \widehat{E}} \delta_{ak} (v_a - \widehat{v_a}) \tag{5.11}$$

$$= \sum_{a \in \widehat{E}} (v_a - \widehat{v_a}) \sum_{k \in P_{ij}} p_k \delta_{ak} \tag{5.12}$$

where $p_k = \frac{v_k}{g_{ij}}$ is probability that user chooses the path k on the trip between origin i and destination j. The $\delta_{ap} \in \{0, 1\}$ is one if the edge a lies on path p else is zero. Let us set

$$p_{ij}^a = \sum_{k \in P_{ij}} p_k \delta_{ak} \tag{5.13}$$

where p_{ij}^a is the probability that the user crosses the edge a on the trip between origin i and destination j. The $\mathbb{P} = \{p_{ij}^a : ij \in W, a \in E\}$ is the assignment matrix. By

substitution (5.13) to (5.11) we obtain the negative value of search direction

$$d_{ij} = \sum_{a \in \widehat{E}} (v_a - \widehat{v_a}) p_{ij}^a \tag{5.14}$$

According to Spiess (1990) the ODM update is performed as

$$g_{ij} = g_{ij}(1 - \lambda d_{ij}) \tag{5.15}$$

where λ is the length of the step. Using the assignment matrix $\mathbb{P}$, the optimal step length can be rewritten as

$$\lambda = \frac{\sum_{a \in \widehat{E}} v'_a (\widehat{v_a} - v_a)}{\sum_{a \in \widehat{E}} (v'_a)^2} \tag{5.16}$$

where

$$v'_a = \frac{\partial v_a}{\partial \lambda} = - \sum_{ij \in W} g_{ij} d_{ij} p_{ij}^a \tag{5.17}$$

5.3.1 The implicit computation of assignment matrix

Let v_a^i be the flow on the edge $a \in E$ that starts in the origin $i \in Z$. The edges where $v_a^i > 0$ creates the acyclic sub-graph called a bush. For more precise definition see Nie (2010). The source node of the edge a is noted as $s(a)$ and the target node $t(a)$.

According to Xie and Nie (2019), the path flow can be computed as

$$v_k = g_{ij} \prod_{a \in k} \phi_a^i \tag{5.18}$$

where

$$\phi_a^i = \frac{v_a^i}{\sum_{e \in I(t(a))} v_e^i} \tag{5.19}$$

where $I(n)$ is a set of incoming edges to the node $n \in V$. The ϕ_a^i represents the proportion of all bush flow inflowing to the target node $t(a)$ of the edge a. The $s(a)$ is the source node of the edge a. Combining (5.13) and (5.19) we have

$$p_{ij}^a = \sum_{k \in P_{ij}} \delta_{ak} \prod_{e \in k} \phi_e^i \tag{5.20}$$

The implicit computation of the assignment matrix p_{ij}^a is based on Breadth First Search (BFS). Let κ_n be a node label representing the probability that a driver crosses the node n on the trip between zones i and j. It follows that

$$p_{ij}^a = \kappa_{t(a)} \phi_a^i \tag{5.21}$$

Algorithm 7: Implicit computation of the assignment matrix. The input is the OD pair $ij \in W$ and the feasible flow v_a^i on bush originated at i.

```
1  o_n = 0   ∀n ∈ V
2  U is the FIFO queue of nodes
3  U.add(j)
4  while |U| > 0 do            // compute the number of outgoing edges o_n
5      n = U.remove()
6      foreach e ∈ I(n) do
7          if v_e^i > 0 then
8              if o_{s(e)} = 0 then
9                  U.add(s(e))
10             o_{s(e)} = o_{s(e)} + 1
11 κ_n = 0   ∀n ∈ V
12 κ_j = 1.0                   // probability at destination node
13 U.add(j)
14 while |U| > 0 do            // determine the p_ij^e for every link in the bush
15     n = U.remove()
16     compute φ_e^i   ∀e ∈ I(n)
17     foreach e ∈ I(n) do
18         if φ_e^i > 0 then
19             p_ij^e = κ_n φ_e^i
20             κ_{s(e)} = κ_{s(e)} + p_ij^e
21             o_{s(e)} = o_{s(e)} − 1
22             if o_{s(e)} = 0 then
23                 U.add(s(e))
```

and

$$\kappa_n = \sum_{a \in O(n)} p_{ij}^a \tag{5.22}$$

where $O(n)$ is a set of outgoing edges from node n. Using the equations (5.21) and (5.22) p_{ij}^a can be determined sequentially for all used edges a corresponding with the OD pair ij.

In Algorithm 7 there is a pseudo-code that computes the assignment matrix for one OD pair. The input of the algorithm is the origin node $i \in Z$, destination node $j \in Z$ and the feasible flow v_a^i on the bush originated at i. For whole assignment matrix, the Algorithm 7 must be run for every OD pair.

First, it is necessary to determine how many outgoing edges from the node n lead to the destination j. The edges with $p_{ij}^a = 0$ do not lead to the destination j. For this purpose, the BFS on the reverse graph is used. The searching starts at the destination node j. In Algorithm 7, the searching is represented by lines 4-10.

The second part of the algorithm sequentially determines p_{ij}^a values. The BFS started at the destination node j is also used for this purpose. The origin and destination node must cross all vehicles so $\kappa_i = \kappa_j = 1.0$. In every edge relaxation p_{ij}^a is determined and the node label $\kappa_{s(a)}$ is updated. The node $s(a)$ is added to the queue only if $\kappa_{s(a)}$ was updated by all outgoing edges leading to the destination. In Algorithm 7, this procedure is represented by lines 14-23.

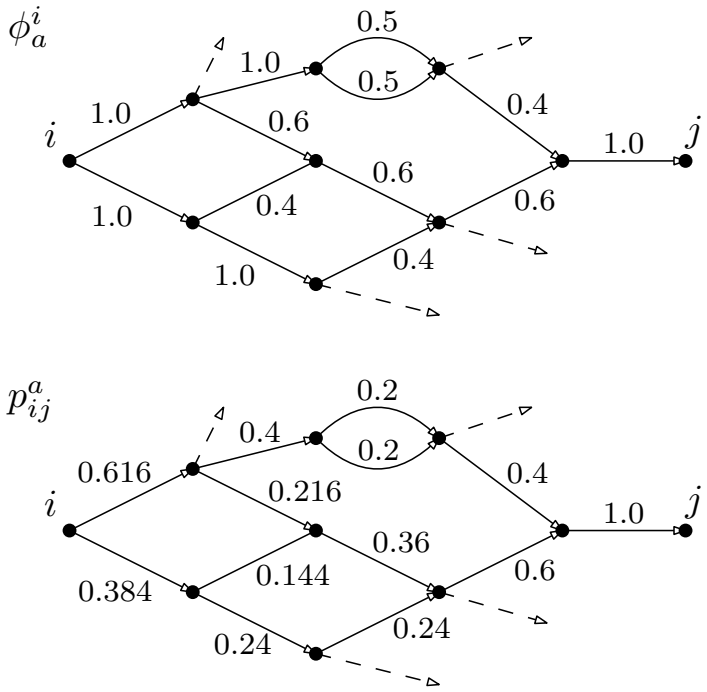


Figure 5.1: Example bush with ϕ_a^i and p_{ij}^a values

In Figure 5.1 you can see an example bush with ϕ_a^i that are computed from the bush flow and the result probability values p_{ij}^a. The dashed lines represent the edges leading to other destinations. These edges are eliminated by the first part of the algorithm.

5.4 Numerical tests

The proposed method and the original method by Spiess (1990) for ODM estimation were implemented in the Java programming language. The B-algorithm (Dial, 2006) for solving UE and the algorithm by Xie and Nie (2019) for determining the MEUE were also implemented in Java. All tests were performed on a laptop with four core processor Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz and with 16 GB RAM. For testing, the real model of Pilsen (the city in the Czech Republic) was used. The model has 9036 edges, 3727 nodes, 316 zones, and 65411 OD pairs. The relative gap for B-algorithm was set to 10^{-10}. For a simulation of the high-congested (HC) situation, the original ODM was multiplied by 2.0.

	number of paths		update runtime [ms]		savings [%]
model	initial	target	explicit	implicit	
Pilsen	66 109	69 179	19 467	9 691	50
Pilsen HC	71 083	124 346	26 241	10 041	62

Table 5.1: Testing results

In Table 5.1 there are results of the tests. The *update runtime* column represents the time that the algorithm spends in the ODM update procedure. The initial number of paths is counted before entropy maximization and the target is measured after entropy maximization. In the case of the original model, there are only 1.06 paths for one OD pair in average. Despite that, the time-savings in the update procedure compared to the original method by Spiess (1990) are 50%. In high-congested case, there are 1.9 paths for one OD pair in average and the time-savings increase to 62%. It follows that the proposed method is suitable in cases, where there is a lot of route choices.

The most time-consuming part of the ODM estimation is the computation of MEUE so the total time savings are only 6% in the HC case. In most state-of-the-art approaches, the entropy maximization is not taken into account. In this case, the time-savings are 17% for the HC model.

5.4.1 Summary

The tests show that the proposed method saves more than 50% of the time that is needed for ODM updates. In the total computation time, the savings are in the order of percent. The most time-consuming part is the computation of MEUE. It would be

interesting to determine how the precision of the MEUE solution influences the quality of ODM estimation and possibly decreases the accuracy of MEUE computation.

Chapter 6

Future research directions

The algorithms presented in this thesis improve the computational efficiency but the dynamic traffic assignment models are still too slow for use with huge road networks. Calibration of these models can take months. From this work, several directions for future research emerged.

It would be nice to use the idea of the piecewise linear/constant-based solution for simultaneous route-and-departure-time dynamic user equilibrium (SRDT-DUE). This could be done in two different ways. First, the equilibrium can be computed using one optimization as is published by Friesz et al. (1993). The second approach is to use the bi-level optimization where the RC-DUE is computed first and after that, the departure times are adjusted.

As you can see in this thesis, dynamic network loading is very time-consuming. The loading method by Raadsen and Bliemer (2019) that is used in this thesis was published recently and therefore there are no optimization techniques developed yet. One of the optimization techniques for repeated computations is marginal sequential updating by Corthout et al. (2014). It would be nice to modify this technique for event-based GLTM. Also, the ε-LCA-BS algorithm can be optimized for repeated computations, e.g., using the ideas by Himpe (2016).

The effectiveness of the algorithms that determine the dynamic equilibrium is much worse compared to the algorithms for static user equilibrium. In this area, some ideas from the static one can be applied, e.g, to try to use the bushes or the technique of paired alternative segments. These ideas were also published by Himpe (2016).

The route choice model based on user equilibrium by Wardrop assumes that every user behaves rationally and chooses the fastest (generally cheapest) path. In reality, this assumption is not fulfilled especially in cases when there are some changes in the route network. Some drivers behave irrationally or just do not know which route is the fastest. This is the big question for the future research.

Bibliography

Aitken, A. C. (1927). Xxv.—on bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305.

Babazadeh, A., Javani, B., Gentile, G., and Florian, M. (2020). Reduced gradient algorithm for user equilibrium traffic assignment problem. *Transportmetrica A: Transport Science*, 16(3):1111–1135. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/23249935.2020.1722279.

Bar-Gera, H. (2002). Origin-Based Algorithm for the Traffic Assignment Problem. *Transportation Science*, 36(4):398–417.

Bar-Gera, H. (2010). Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological*, 44(8-9):1022–1046.

Batz, G. V., Geisberger, R., Sanders, P., and Vetter, C. (2013). Minimum time-dependent travel times with contraction hierarchies. *Journal of Experimental Algorithmics*, 18:1.1–1.43.

Bliemer, M. C. J. and Raadsen, M. P. H. (2019). Continuous-time general link transmission model with simplified fanning, Part I: Theory and link model formulation. *Transportation Research Part B: Methodological*, 126:442–470.

Cascetta, E. (1984). Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator. *Transportation Research Part B: Methodological*, 18(4-5):289–299.

Corthout, R., Flötteröd, G., Viti, F., and Tampère, C. M. J. (2012). Non-unique flows in macroscopic first-order intersection models. *Transportation Research Part B: Methodological*, 46(3):343–359.

Corthout, R., Himpe, W., Viti, F., Frederix, R., and Tampère, C. M. (2014). Improving the efficiency of repeated dynamic network loading through marginal simulation. *Transportation Research Part C: Emerging Technologies*, 41:90–109.

Dafermos, S. (1980). Traffic Equilibrium and Variational Inequalities. *Transportation Science*, 14(1):42–54.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287.

Daganzo, C. F. (1995). The cell transmission model, part II: Network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93.

Dean, B. C. (1999). *Continuous-Time Dynamic Shortest Path Algorithms*. PhD thesis, Massachusetts Institute of Technology.

Dean, B. C. (2004). Shortest paths in FIFO time-dependent networks: Theory and algorithms. *Rapport technique, Massachusetts Institute of Technology*.

Dehne, F., Omran, M. T., and Sack, J.-R. (2009). Shortest paths in time-dependent FIFO networks using edge load forecasts. In *Proceedings of the Second International Workshop on Computational Transportation Science*, pages 1–6. ACM.

Dehne, F., Omran, M. T., and Sack, J.-R. (2012). Shortest Paths in Time-Dependent FIFO Networks. *Algorithmica*, 62(1-2):416–435.

Dey, S. S. and Fricker, J. D. (1994). Bayesian updating of trip generation data: combining national trip generation rates with local data. *Transportation*, 21(4):393–403.

Dial, R. B. (2006). A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B: Methodological*, 40(10):917–936.

Ding, B., Yu, J. X., and Qin, L. (2008). Finding time-dependent shortest paths over large graphs. page 205. ACM Press.

Facchinei, F. and Pang, J.-S., editors (2004). *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer Series in Operations Research and Financial Engineering. Springer New York, New York, NY.

Flötteröd, G. and Rohde, J. (2011). Operational macroscopic modeling of complex urban road intersections. *Transportation Research Part B: Methodological*, 45(6):903–922.

Forschungsgesellschaft für Strassen und Verkehrswesen, K. (2001). Handbuch für die Bemessung von Strassenverkehrsanlagen. Standard, FSGV Verlag GmbH.

Foschini, L., Hershberger, J., and Suri, S. (2014). On the Complexity of Time-Dependent Shortest Paths. *Algorithmica*, 68(4):1075–1097.

Friesz, T. L., Bernstein, D., Smith, T. E., Tobin, R. L., and Wie, B.-W. (1993). A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, 41(1):179–191.

Friesz, T. L., Kim, T., Kwon, C., and Rigdon, M. A. (2011). Approximate network loading and dual-time-scale dynamic user equilibrium. *Transportation Research Part B: Methodological*, 45(1):176–207.

Geisberger, R. (2010). Engineering Time-dependent One-To-All Computation. *arXiv:1010.0809 [cs]*. arXiv: 1010.0809.

Geisberger, R. and Sanders, P. (2010). Engineering time-dependent many-to-many shortest paths computation. In *OASIcs-OpenAccess Series in Informatics*, volume 14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Gentile, G. (2014). Local User Cost Equilibrium: a bush-based algorithm for traffic assignment. *Transportmetrica A: Transport Science*, 10(1):15–54.

Gentile, G. (2015). Using the General Link Transmission Model in a Dynamic Traffic Assignment to Simulate Congestion on Urban Networks. *Transportation Research Procedia*, 5:66–81.

Gentile, G. (2016). Solving a Dynamic User Equilibrium model based on splitting rates with Gradient Projection algorithms. *Transportation Research Part B: Methodological*, 92:120–147.

Gibb, J. (2011). Model of Traffic Flow Capacity Constraint through Nodes for Dynamic Network Loading with Queue Spillback. *Transportation Research Record: Journal of the Transportation Research Board*, 2263(1):113–122.

Han, D. and Lo, H. K. (2004). Solving non-additive traffic assignment problems: A descent method for co-coercive variational inequalities. *European Journal of Operational Research*, 159(3):529–544.

Han, K., Eve, G., and Friesz, T. (2018). Computing Dynamic User Equilibria on Large-Scale Networks: From Theory to Software Implementation. *arXiv:1810.00777 [math]*. arXiv: 1810.00777.

Han, K., Friesz, T. L., Szeto, W. Y., and Liu, H. (2015). Elastic demand dynamic network user equilibrium: Formulation, existence and computation. *Transportation Research Part B: Methodological*, 81:183–209.

Han, K., Piccoli, B., and Friesz, T. L. (2016). Continuity of the path delay operator for dynamic network loading with spillback. *Transportation Research Part B: Methodological*, 92:211–233.

Himpe, W. (2016). *Integrated algorithms for repeated dynamic traffic assignments*. PhD thesis, KU Leuven, Leuven.

Himpe, W., Corthout, R., and Tampère, M. C. (2016). An efficient iterative link transmission model. *Transportation Research Part B: Methodological*, 92:170–190.

Hoogendoorn, S. P. and Bovy, P. H. L. (2001). State-of-the-art of vehicular traffic flow modelling. 215:21.

Huang, H.-J. and Lam, W. H. (2002). Modeling and solving the dynamic user equilibrium route and departure time choice problem in network with queues. *Transportation Research Part B: Methodological*, 36(3):253–273.

Imai, H. and Iri, M. (1986). An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing*, 9(3):159–162.

Jabari, S. E. (2016). Node modeling for congested urban road networks. *Transportation Research Part B: Methodological*, 91:229–249.

Jang, W., Ran, B., and Choi, K. (2005). A discrete time dynamic flow model and a formulation and solution method for dynamic route choice. *Transportation Research Part B: Methodological*, 39(7):593–620.

Kolovský, F. and Kolingerová, I. (2021). Origin-destination matrix estimation using bush-based user equilibrium algorithms. In Gervasi, O., Murgante, B., Misra, S., Garau, C., Blečić, I., Taniar, D., Apduhan, B. O., Rocha, A. M. A. C., Tarantino, E., and Torre, C. M., editors, *Computational Science and Its Applications – ICCSA 2021*, pages 285–294, Cham. Springer International Publishing.

Kolovský, F., Ježek, J., and Kolingerová, I. (2019a). The e-approximation of the Label Correcting Modification of the Dijkstra's Algorithm. In *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management - Volume 1: GISTAM,*, pages 26–32. SciTePress.

Kolovský, F., Ježek, J., and Kolingerová, I. (2019b). The ε-Approximation of the Time-Dependent Shortest Path Problem Solution for All Departure Times. *ISPRS International Journal of Geo-Information*, 8(12):538. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

Kolovský, F., Kolingerová, I., and Ježek, J. (2019c). Algorithms in transportation: The State of the Art and Concept of PhD Thesis.

Konno, H. and Kuno, T. (1988). Best piecewise constant approximation of a function of single variable. *Operations Research Letters*, 7(4):205–210.

Lebacque, J.-P. (2005). First-order macroscopic traffic flow models: Intersection modeling, network modeling. In *Transportation and Traffic Theory. Flow, Dynamics and Human Interaction. 16th International Symposium on Transportation and Traffic TheoryUniversity of Maryland, College Park*.

LeBlanc, L. J., Helgason, R. V., and Boyce, D. E. (1985). Improved Efficiency of the Frank-Wolfe Algorithm for Convex Network Programs. *Transportation Science*, 19(4):445–462.

Lighthill, M. J. and Whitham, G. B. (1955). On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345.

Lo, H. K. and Szeto, W. Y. (2002). A cell-based variational inequality formulation of the dynamic user optimal assignment problem. *Transportation Research Part B: Methodological*, 36(5):421–443.

Long, J., Huang, H.-J., Gao, Z., and Szeto, W. Y. (2013). An Intersection-Movement-Based Dynamic User Optimal Route Choice Problem. *Operations Research*, 61(5):1134–1147.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Lundgren, J. T. and Peterson, A. (2008). A heuristic for the bilevel origin–destination-matrix estimation problem. *Transportation Research Part B: Methodological*, 42(4):339–354.

Maher, M. (1983). Inferences on trip matrices from observations on link volumes: a bayesian statistical approach. *Transportation Research Part B: Methodological*, 17(6):435–447.

Nezamuddin, N. and Boyles, S. D. (2015). A Continuous DUE Algorithm Using the Link Transmission Model. *Networks and Spatial Economics*, 15(3):465–483.

Nie, Y. M. (2010). A class of bush-based algorithms for the traffic assignment problem. *Transportation Research Part B: Methodological*, 44(1):73–89.

Omran, M. and Sack, J.-R. (2014). Improved approximation for time-dependent shortest paths. In *International Computing and Combinatorics Conference*, pages 453–464. Springer.

Orda, A. and Rom, R. (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625.

Perederieieva, O., Ehrgott, M., Raith, A., and Wang, J. Y. (2015). A framework for and empirical study of algorithms for traffic assignment. *Computers & Operations Research*, 54:90–107.

Raadsen, M. P. H. and Bliemer, M. C. J. (2019). Continuous-time general link transmission model with simplified fanning, Part II: Event-based algorithm for networks. *Transportation Research Part B: Methodological*, 126:471–501.

Raadsen, M. P. H., Bliemer, M. C. J., and Bell, M. G. H. (2016). An efficient and exact event-based algorithm for solving simplified first order dynamic network loading problems in continuous time. *Transportation Research Part B: Methodological*, 92:191–210.

Ran, B., Hall, R. W., and Boyce, D. E. (1996). A link-based variational inequality model for dynamic departure time/route choice. *Transportation Research Part B: Methodological*, 30(1):31–46.

Richards, P. I. (1956). Shock waves on the highway. *Operations Research*, 4(1):42–51.

Rostami Nasab, M. and Shafahi, Y. (2020). Estimation of origin–destination matrices using link counts and partial path data. *Transportation*, 47(6):2923–2950.

Smith, M. (1979). The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, 13(4):295–304.

Smits, E.-S., Bliemer, M. C. J., Pel, A. J., and van Arem, B. (2015). A family of macroscopic node models. *Transportation Research Part B: Methodological*, 74:20–39.

Song, W., Han, K., Wang, Y., Friesz, T., and del Castillo, E. (2017). Statistical metamodeling of dynamic network loading. *Transportation Research Procedia*, 23:263–282.

Spiess, H. (1987). A maximum likelihood model for estimating origin-destination matrices. *Transportation Research Part B: Methodological*, 21(5):395–412.

Spiess, H. (1990). A gradient approach for the OD matrix adjustment problem. 1:2.

Steffensen, J. (1933). Remarks on iteration. *Scandinavian Actuarial Journal*, 1933(1):64–72.

Strasser, B. (2017). Dynamic Time-Dependent Routing in Road Networks Through Sampling. In *OASIcs-OpenAccess Series in Informatics*, volume 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Szeto, W. and Lo, H. K. (2004). A cell-based simultaneous route and departure time choice model with elastic demand. *Transportation Research Part B: Methodological*, 38(7):593–612.

Szeto, W. Y. and Lo, H. K. (2006). Dynamic traffic assignment: properties and extensions. *Transportmetrica*, 2(1):31–52.

Tampère, C. M., Corthout, R., Cattrysse, D., and Immers, L. H. (2011). A generic class of first order node models for dynamic macroscopic simulation of traffic flows. *Transportation Research Part B: Methodological*, 45(1):289–309.

Tian, L.-J., Huang, H.-J., and Gao, Z.-Y. (2012). A Cumulative Perceived Value-Based Dynamic User Equilibrium Model Considering the Travelers' Risk Evaluation on Arrival Time. *Networks and Spatial Economics*, 12(4):589–608.

TP188 (2018). Posouzení kapacity všech druhů křižovatek a úseků pozemních komunikací. Standard, Ministry of Transport of Czech Republic.

Transportation Networks for Research Core Team (2021). Transportation networks for research. https://github.com/bstabler/TransportationNetworks.

Ukkusuri, S. V., Han, L., and Doan, K. (2012). Dynamic user equilibrium with a path based cell transmission model for general traffic networks. *Transportation Research Part B: Methodological*, 46(10):1657–1684.

van der Gun, J. P. T., Pel, A. J., and van Arem, B. (2017). Extending the Link Transmission Model with non-triangular fundamental diagrams and capacity drops. *Transportation Research Part B: Methodological*, 98:154–178.

van Zuylen, H. J. and Branston, D. M. (1982). Consistent link flow estimation from counts. *Transportation Research Part B: Methodological*, 16(6):473–476.

Van Zuylen, H. J. and Willumsen, L. G. (1980). The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological*, 14(3):281–293.

Wardrop, J. G. (1952). Road paper. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362.

Wright, M. A., Gomes, G., Horowitz, R., and Kurzhanskiy, A. A. (2017). On node models for high-dimensional road networks. *Transportation Research Part B: Methodological*, 105:212–234.

Wu, X. and Liu, H. X. (2011). A shockwave profile model for traffic flow on congested urban arterials. *Transportation Research Part B: Methodological*, 45(10):1768–1786.

Xie, J. and Nie, Y. M. (2019). A New Algorithm for Achieving Proportionality in User Equilibrium Traffic Assignment. *Transportation Science*, 53(2):566–584.

Xie, J., Nie, Y. M., and Liu, X. (2018). A Greedy Path-Based Algorithm for Traffic Assignment. *Transportation Research Record*, 2672(48):36–44. Publisher: SAGE Publications Inc.

Xie, J. and Xie, C. (2016). New insights and improvements of using paired alternative segments for traffic assignment. *Transportation Research Part B: Methodological*, 93:406–424.

Yahyamozdarani, R., Himpe, W., and Tampère, C. M. (2019). A continuum approach for modeling signalized nodes in dynamic traffic assignment. In *8th Symposium of the European Association for Research in Transportation (hEART 2019), Date: 2019/09/04-2019/09/06, Location: Budapest*.

Yperman, I. (2007). *The Link Transmission Model for Dynamic Network Loading*. Katholieke Universiteit Leuven, Belgium.

Appendix A

Professional Activities

A.1 Publications

A.1.1 Journals

Kolovský, F., Ježek, J., Kolingerová, I. (2019). The ε-Approximation of the Time-Dependent Shortest Path Problem Solution for All Departure Times. ISPRS International Journal of Geo-Information, MDPI 8(12), 538.

Potužák, T., **Kolovský, F.** (2021). Parallelization of the B Static Traffic Assignment Algorithm. Ain Shams Engineering Journal.

A.1.2 International Conferences

Kolovský, F., Kolingerová, I. (2021). Origin-Destination Matrix Estimation Using Bush-Based User Equilibrium Algorithms. In International Conference on Computational Science and Its Applications (pp. 285-294). Springer, Cham.

Jedlička K., Beran, D., Martolos J., **Kolovský F**., Kepka M., Mildorf T., Sháněl, J. (2020). Traffic modelling for the smart city of Pilsen. In 8ICCGIS PROCEEDINGS VOL.1 (pp. 510-520). Chr. Smirnenski Blvd. Sofia, Bulgaria, 2020: Bulgarian Cartographic Association, 2020.

Kolovský, F., Ježek, J., Kolingerová, I. (2019). The ε-approximation of the Label Correcting Modification of the Dijkstra's Algorithm. In 5th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM) (pp. 26-32). Heraklion, Crete - Greece

Kolovský, F., Ježek, J., Kolingerová, I. (2018). The Origin-Destination matrix estima-

tion for large transportation models in an uncongested network. In 2018 International Conference on Mathematical Applications (pp. 17 - 22). Madeira - Portugal.

A.1.3 Under review

Kolovský, F., Kolingerová, I. The piecewise constant/linear solution for dynamic user equilibrium.

Kolovský, F., Kolingerová, I., Martolos, J., Potužák, T. Capacity based first order node model for dynamic traffic loading.

A.2 Participation in Scientific Projects

- Grant agreement ID 296282: Plan4business - A service platform for aggregation, processing and analysis of urban and regional planning data, H2020, 2013-2014.
- Grant agreement ID 620533: OpenTransportNet - Spatially Referenced Data Hubs for Innovation in the Transport Sector, H2020, 2014-2017.
- Grant agreement ID 769608: PoliVisu - Policy Development based on Advanced Geospatial Data Analytics and Visualisation, H2020, 2017-2020.
- Grant agreement ID 870697: DUET - Digital Urban European Twins for smarter decision making, H2020, 2019-now.
- Grant agreement ID 883522: S4AllCities - Smart Spaces Safety and Security for All Cities, H2020, 2020-now.
- CK01000096: TRAFFO - Innovative Approaches to Mathematical Traffic Modelling for Sustainable Development of Cities and Regions, The Technology Agency of the Czech Republic, 2019-now.
- SGS-2016-004: Application of Mathematics and Informatics in Geomatics III University of West Bohemia (UWB).
- SGS-2019-015: Application of Mathematics and Informatics in Geomatics IV University of West Bohemia (UWB).

A.2.1 Project publications

Jedlička K., Hájek P., Ježek J., **Kolovský F**., Beran D., Mildorf T., Charvát K., Kozhukh D., Martolos J., Šťastný J., (2017). Otevřená dopravní mapa pro Evropu. Symposium GIS Ostrava 2017 Geoinformatika v pohybu Sborník.

Jedlička K., Hájek P., Ježek J., **Kolovský F**., Mildorf T., Charvát K., Kozhukh D., Martolos J., Šťastný J., Beran D., (2016). Open Transport Map: open, harmonized dataset or road network. AutoCarto 2016 Proceedings (pp. 72-84).

Jedlička, K., Hájek, P., Cada, V., Martolos, J., Šťastný, J., Beran, D., **Kolovský F**., Kozhukh, D. (2016). Open transport map—Routable OpenStreetMap. In 2016 IST-Africa Week Conference (pp. 1-11). IEEE.

Jedlička, K., Ježek J., Kepka M., Hájek P., Mildorf T., **Kolovský F**., Beran D. (2015). Dynamic Visualization of Volume of Traffic. Papers ICC (pp. 1-13).

A.3 Talks

Origin-destination matrix estimation using bush-based algorithms. *The 21st International Conference on Computational Science and its Applications (ICCSA 2021).* Cagliari, Italy. 16.9.2021

Traffic modeling, monitoring and visualization. *Norwegian University of Science and Technology (NTNU).* Trondheim, Norway. 12.6.2019

The ε-approximation of label correcting modification of Dijkstra's algorithm. *Joint conference of ISAF & Geomatics in Projects & Plan4All in Pilsen.* Plzeň. 4.10.2018

Rozdíly modelování dopravy v urbanizovaných oblastech a mimo ně. *Joint conference of ISAF & Geomatics in Projects & Plan4All in Pilsen.* Státní zámek Kozel, Štáhlavy. 5.10.2017

Restricted Areas Obeying Path Search Algorithm without Preprocessing. *23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015).* Seattle, Washington, USA. 3.11.2015