Technische Universität Berlin

Bachelorarbeit

Nash Flows over Time
- Models and Computation -

eingereicht von:	Max Zimmer
	Matrikelnummer: 352801
eingereicht am:	16.08.2017
Fakultät:	Fakultät II
	Mathematik und Naturwissenschaften
	Institut für Mathematik
Erstprüfer:	Prof. Dr. Martin Skutella
Zweitprüfer:	Prof. Dr. Thorsten Koch
Betreuer:	Leon Sering

Eidesstattliche Erklärung

Die selbstständige und eigenhändige Anfertigung versichert an Eides statt

Berlin, den 16.08.2017

_____

Max Zimmer

Abstract

We consider Nash flows over time from a computational point of view. Flows over time generalize the concept of static s-t-flows by considering a continuous stream of particles, flowing through a network, defined by flow rates on edges on which queues can build up and vanish, inducing temporal delays whilst traveling from s towards t. The particular interest of this thesis is the study of dynamic flows which further form a Nash equilibrium, that is every flow particle is treated as a player and acts selfishly by trying to minimize its travel time from the source to the sink. The only existing approach of computing Nash flows over time relies on the concept of thin flows with resetting, a special class of static s-t-flows. For the time being not much is known about the efficient computation of thin flows. After rigorously introducing a mathematical model for flows over time and presenting existing results regarding the characterization of Nash equilibria, we investigate and discuss models allowing the computation of thin flows. We prove the correctness of newly introduced approaches and present an implementation, which computes and visualizes Nash flows over time.

Wir betrachten dynamische Nash Flüsse. Das Hauptaugenmerk liegt dabei auf deren Berechnung. Dynamische Flüsse verallgemeinern das Konzept der statischen s-t-Flüsse indem nun ein kontinuierlicher Fluss durch ein Netzwerk betrachtet wird, welcher durch Flussraten auf den Kanten definiert ist. Auf den Kanten des Netzwerks können sich Warteschlangen auf- und abbauen, welche zeitliche Verzögerungen der Flusspartikel auf ihrem Weg von der Quelle s zur Senke t zur Folge haben. Schwerpunkt der vorliegenden Arbeit ist die Analyse von besonderen dynamischen Flüssen, welche aus spieltheoretischer Betrachtungsweise ein Nash-Gleichgewicht darstellen. Jedes Flussteilchen wird als eigennütziger Spieler betrachtet und versucht seine Kosten, d.h. die benötigte Flusszeit von s nach t, zu minimieren. Der derzeit einzige Ansatz zur Berechnung von dynamischen Nash Flüssen beruht auf einer Klasse spezieller statischer Flüsse, sogenannte "thin flows with resetting". Eine der Problematiken besteht darin, dass Methoden zu deren effizienter Berechnung zum jetzigen Zeitpunkt lediglich partiell erforscht sind. Nach einer ausführlichen Erläuterung eines mathematischen Modellierungsansatzes dynamischer Flüsse sowie einer Rekapitulation bestehender Resultate bezüglich charakterisierender Eigenschaften der Nash Flüsse, folgt eine Untersuchung möglicher Modelle zur Berechnung von "thin flows". Wir beweisen die Korrektheit von neuen Lösungsansätzen und stellen eine Implementation vor, welche sich der Berechnung sowie graphischen Visualisierung dynamischer Nash Flüsse widmet.

Contents

1 Introduction

In a typical Combinatorial Optimization course students are confronted with network flow problems, that is sending an amount of flow from a source s through a graph G towards a sink t. The most common intuition behind this concept, often conveyed by the lecturer, is the following: flow networks allow modeling problems like sending a maximum amount of water or oil through a pipeline network, efficient routing of packets through computer networks or minimum cost shipment of some entitity. The essence of all of these problems lies in the desire of moving something from one point to another in an efficient manner. While flow problems and algorithms have, since their formulation, not only gained huge popularity but also provided a tool kit to solve many problems, they often fail to adequately model real world examples. In many cases they lack the capability to model a crucial parameter: **time**. Consider for example the traffic flow in packet switching networks. Indubitably, the number or size of all packets on a certain channel (interpretable as an edge in a computer network) varies over time. Furthermore, packets travelling through edges take a certain amount of time to reach their destination, neither captured by a static flow model. Hence, static flows do not provide a model to sufficiently handle these problems. In this thesis, we thus focus on a more complex type of flows.

Flows which incorporate a time parameter, associating with each edge a time-dependent flow value, are called *flows over time* or *dynamic flows*. Each edge is assigned a constant time that is needed to traverse it, namely its *free flow transit time*. Furthermore each edge has a *capacity* (sometimes called *service rate*), which acts as an upper bound on the amount of flow traveling through the edge per time unit. Flow is injected at a source s and travels towards a sink t experiencing a delay induced by the transit times of the edges on the paths from s to t. Routing flow through certain edges at a certain time has then obviously a direct effect on later routing decisions, having to obey the capacity constraints.

Ford and Fulkerson [5] introduced this basic model in 1962 by raising (and answering) the question how to compute the maximal amount of flow that one can send from s to t in a given time horizon T (known as the *maximum flow over time* problem). Gale [7] picked up the model in his publication "Transient Flows in Networks" and showed the existence of so-called *earliest arrival flows*. These are flows over time maximizing the amount of flow reaching t at all time steps simultaneously. While these models were considered in a discrete time setting, at the end of the twentieth century the transition to continuous time models was made (see e.g. [4]). Skutella covered the evolution of the field of flows over time in [12], introducing not only a variety of related problems and solutions but also their historical evolvement.

This thesis, however, considers a flow over time model having a peculiarity. If more flow wants to enter an edge than the edge capacity allows, the flow waits in a queue, whose size and consequently the induced queuing delay are only dependent on previous flow on said edge. Hence, the time needed to travel

from s towards t is not only affected by the (constant) free flow transit times of edges along an s-t-path, but also by their queuing delays, regulated by flow that went through these edges at an earlier point in time. The model, called *fluid queue model* or *deterministic queuing model*, attracted the interest of both the traffic and game-theory researching communities. From a game-theoretical perspective each flow particle in time can be seen as a player, having to choose an s-t-path to travel along. The players objective is to reduce his costs, that is, the time needed to travel to t. We emphasize the complexity of the problem of finding the shortest s-t-path. A player starting at s has to anticipate queues on edges, which may build up, up to the time of his arrival, but might be empty at his departure. Thus, the propagation of flow across the network in a way such that every particle at a time minimizes its cost, is a highly non-trivial task. This state is achieved if every particle or player acts selfishly, i.e., chooses a shortest s-t-path. Such a flow is in fact a (weak) Nash equilibrium and thus called a Nash flow over time. A Nash equilibrium occurs if no player has the incentive to change his strategy, that is his travel path.

Koch and Skutella [8] introduced the concept of *thin flows with resetting*, a specific class of static s-t-flows, whose concatenation can be seen as a characterization of a Nash flow over time (given the injection of a constant inflow at s). Furthermore, the authors describe a method for the computation of flows over time, which is based on the computation of thin flows with resetting. On this basis, Cominetti, Correa and Larré ([1], [3]) presented a finite but exponential time algorithm to compute thin flows. The motivation behind this thesis is to discuss and improve existing models related to the problem of finding a thin flow and consequently, a Nash flow over time.

The thesis is subdivided in several sections. In section 2 we formally introduce the concepts of flows over time and thin flows with resetting. Furthermore, we discuss existing results regarding the characterization of aforementioned thin flows as well as Nash flows over time, following closely the publications of Cominetti, Correa and Larré ([1], [3]) and Koch and Skutella [8]. Section 3 describes Kochs and Skutellas method to compute Nash flows over time. As thin flows with resetting play an important role in said method, we dedicate section 4 entirely to the problem of computing such flows. We dissolve an ambiguity regarding an existing method (cp. [1]) and present two new, albeit similar, approaches. In the last section we present an implementation of the discussed methods to compute Nash flows over time, embedded in a user interface allowing the creation and analysis of flow networks as well as an animated visualization of the flow streams. Finally, Appendix A contains auxiliary definitions and results from measure theory and functional analysis, which are used in this thesis.

2 A flow over time model and Nash flows

In this section we stick to the notation of Cominetti et al., although many of the results have originally been stated by Koch and Skutella. The former reproduced many of the results in a formally more precise setting. While this section more or less follows the structure of [1], we occasionally compare both publications. It is worth noting that we often use measure theoretical as well as functional analysis terms. For a proper explanation we recommend to first take a glance at the appendix.

2.1 Definition and feasibility

Consider a network given by a directed graph $G = (V, E)$ with a source and a sink, $s, t \in V$. To each directed edge $e = vw \in E$ from v to w we assign two values, namely the *free flow transit time* $\tau_e \geq 0$ and the *capacity* $\nu_e > 0$. While the former is a constant delay that flow, traveling through e, is experiencing, the latter represents the maximum amount of flow per time unit that can leave the queue and in particular, the edge. To simplify the problem, we assume that there is no cycle C in G such that $\sum_{e \in C} \tau_e = 0$.

At all times $\theta \in \mathbb{R}$, flow is being injected into the network at source s at a time-dependent rate $u(\theta)$, given by a locally integrable (cp. Appendix) inflow rate function $u : \mathbb{R} \to \mathbb{R}_{\geq 0}$, which is vanishing on the negative axis, that is $u(\theta) = 0$ for almost all $\theta < 0$. The function assigns to each time $\theta \in \mathbb{R}$ a non-negative amount of flow, which is then routed from s towards t. It thus stands to reason to define the amount of flow that has been injected into the network at s up to a time θ, the cumulative inflow, as the integral over u up to θ. Formally, we define the function

$$\begin{aligned} U : \mathbb{R} &\to \mathbb{R}_{\geq 0} \\ \theta &\mapsto \int_0^\theta u(\xi) d\xi. \end{aligned}$$

Remark 2.1.
Using Lemma A.1 it follows immediately that U is locally absolutely continuous, that is $U \in AC_{loc}(\mathbb{R})$.

Without loss of generality we assume that each node $v \in V$ is reachable from s, i.e. there is an s-v-path in G. If there exists a node that violates said assumption, it is of no harm whatsoever to simply remove this node. The problem of finding a Nash flow over time remains unaffected thereof.

From a game theoretical perspective, the inflow can be interpreted as an infinite amount of infinitesimally small particles or non-identical players with the intention of reaching the sink t in the shortest possible time. We thus identify each player with the point in time θ where its particle starts to flow towards the sink. As Koch and Skutella [8] illustrated, the game theoretical problem can be formulated using a cost function of the time θ, assigning a cost to each player and its strategy, an s-t-path P. A Nash equilibrium

occurs exactly when no player has the incentive to unilaterally deviate from its strategy, that is changing its "to-be-routed-along-s-t-path" in order to lower the travel time. This happens when each player chooses the shortest s-t-path. As opposed to static routing games computing a shortest path is not easy at all. The problem is the following: the travel time a player at θ has to experience does not only depend on the free flow transit time τ_e for each edge e on the chosen path P. It is also dependent on the so-called *latency* of said edges. In other words, the time needed to travel to an edge e depends on the edges which have been traveled prior to that, whose travel time also depends on the amount of flow that entered (and left) the edges earlier.

Definition 2.1. A *flow over time* $f = (f^+, f^-)$ is a pair of families of non-negative locally integrable flow rate functions, which vanish on the negative axis, assigning to each edge $e \in E$ a time-dependent inflow- $f_e^+ : \mathbb{R} \to \mathbb{R}_{\geq 0}$ and outflow-rate $f_e^- : \mathbb{R} \to \mathbb{R}_{\geq 0}$.

For each time $\theta \in \mathbb{R}$ and edge $e \in E$, $f_e^+(\theta)$ and $f_e^-(\theta)$ specify the amount of flow (per time unit) entering and leaving e at θ, respectively. Similarly to the inflowrate of our network, it makes sense to define the locally absolutely continuous and nondecreasing cumulative in- and outflow functions of an edge e, too:

$$F_e^+ : \mathbb{R} \to \mathbb{R}_{\geq 0} \qquad\qquad F_e^- : \mathbb{R} \to \mathbb{R}_{\geq 0}$$
$$\theta \mapsto \int_0^\theta f_e^+(\xi)d\xi \qquad\qquad \theta \mapsto \int_0^\theta f_e^-(\xi)d\xi.$$

For a time θ, $F_e^+(\theta)$ can be interpreted as the amount of flow that has entered e up to θ, while $F_e^-(\theta)$ is the amount of flow having traveled completely through e before θ.
Before we can start with a characterization of Nash flows over time, it is necessary to impose a couple of constraints on flows over time to ensure the feasibility of our modelling. Similarly to the capacity constraints of static s-t-flows, it seems natural to introduce an upper bound on the flow over time on an edge e. We strive for the establishment of a model having queues at the beginning of the edges. With this is mind, it becomes obvious that we have to bound the outflowrate of our edges as follows, making sure that the per time unit flow out of an edge cannot be bigger than the capacity.

$$f_e^-(\theta) \leq \nu_e \text{ for almost all } \theta \in \mathbb{R} \tag{1}$$

Furthermore, wanting that a queue builds up on an edge and not on nodes, we enforce the non-disruptive routing of flow directly through nodes by the following flow conservation constraint, having to hold for almost all times θ:

$$\sum_{e \in \delta^+(v)} f_e^+(\theta) - \sum_{e \in \delta^-(v)} f_e^-(\theta) = \begin{cases} u(\theta) & \text{if } v = s \\ 0 & \text{if } v \in V \setminus \{s, t\}. \end{cases} \tag{2}$$

Obviously, the outflow of an edge should in some way depend on previous inflows. It makes no sense to allow a definition of f_e^- in a way such that flow

is leaving e before even having entered e. Thus, we enforce that the amount of flow having left e after traveling through it can by no means be larger than the amount of flow having entered it. Mathematically, this leads to the following condition:

$$F_e^+(\theta) - F_e^-(\theta + \tau_e) \geq 0 \text{ for all } \theta \in \mathbb{R}. \tag{3}$$

Remark 2.2.
While Koch and Skutella [8] call an s-t-flow over time f *feasible* only if all three equations hold, Cominetti et al. [1] use a slightly different modelling approach by calling f feasible if it merely satisfies the flow conservation constraint (2). However, both approaches are equivalent, as the equations (1) and (3), which have been omitted in the definition of feasibility by Cominetti et al., are a direct implication of a further assumption needed in both papers, namely the assumption that queues *operate at capacity*. A proof that both approaches are equivalent can be found in [1].

2.2 Queue sizes and delays

The flow dynamic of an edge $e = vw$ is modeled as follows. A particle departing from v should first queue up, in case a queue exists. The service rate ν_e determines the rate at which the queue is served. After leaving the queue, the particle should travel to w in τ_e units of time. Figure 2.1 visualizes the modelling approach.

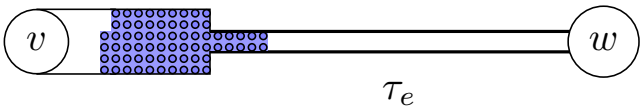


Figure 2.1: An edge $e = vw$. Before traveling towards w and experiencing a transit time τ_e, particles wait in the queue. The service rate ν_e can be interpreted as the width of the edge, controlling the amount of flow that can leave the queue per time unit.

The queue size of e at time θ can thus be explicitly stated. Clearly, the amount of flow that entered e up to θ is given by the cumulative inflow $F_e^+(\theta)$. The amount of flow that has been routed through e but has already left the queue at time θ is therefore given by the amount of flow that has reached w up to the sum of θ and the free flow transit time τ_e, that is $F_e^-(\theta + \tau_e)$.

Definition 2.2. For a given flow over time f and time $\theta \in \mathbb{R}$ we define the *queue size* $z_e(\theta)$ as follows.

$$z_e(\theta) := F_e^+(\theta) - F_e^-(\theta + \tau_e)$$

It is worth noting that by Lemma A.2 we have $z_e \in AC_{loc}(\mathbb{R})$. Furthermore, every $AC_{loc}(\mathbb{R})$ function is continuous. The queue size is nonnegative by assumption (3).
Now, as we have introduced the concept of queues, we are able to make the crucial aforementioned assumption, namely that queues *operate at capacity*.

Definition 2.3. We say that queues *operate at capacity* if for all edges $e \in E$ the following equation holds for almost all $\theta \in \mathbb{R}$.

$$f_e^-(\theta + \tau_e) = \begin{cases} \nu_e & \text{if } z_e(\theta) > 0 \\ \min\{f_e^+(\theta), \nu_e\} & \text{otherwise} \end{cases} \tag{4}$$

Thus, the outflow rate of an edge is defined by its inflow rate (z_e can be expressed without f_e^-, see [1]).
To simplify later computations, we observe that, using the fact that queues operate at capacity, that f_e^+ vanishes on the negative axis and that $z_e(\xi) = 0$ for $\xi < 0$, the following equality holds for all $\theta \in \mathbb{R}$.

$$\begin{aligned} z_e(\theta) = F_e^+(\theta) - F_e^-(\theta + \tau_e) &= \int_0^\theta f_e^+(\xi)d\xi - \int_0^{\theta+\tau_e} f_e^-(\xi)d\xi \\ &= \int_0^\theta f_e^+(\xi)d\xi - \int_0^\theta f_e^-(\xi + \tau_e)d\xi \\ &= \int_0^\theta f_e^+(\xi) - f_e^-(\xi + \tau_e)d\xi \end{aligned} \tag{5}$$

Now, having explicitly stated the queue size at a time θ, allows us to derive a formulation of the queue waiting time or queue delay. A particle arriving at e at θ should naturally have to wait in the queue the minimal amount of time units needed for the entire flow in the queue to leave the queue. As queues operate at capacity, this motivates the following definition.

Definition 2.4. Given a time $\theta \in \mathbb{R}$ we define the *queue delay* $q_e(\theta)$ of e at θ as follows.

$$q_e(\theta) := \frac{z_e(\theta)}{\nu_e} \tag{6}$$

This definition suits our model. Clearly, (3) implies that the queue delay is nonnegative. Observe that the queue remains nonempty on the time interval $W_e^\theta := [\theta, \theta + q_e(\theta)[$ on which a particle entering e at θ has to wait in the queue. Let $q_e(\theta) > 0$ and $\theta' \in W_e^\theta \neq \emptyset$. Using Equation 5 and the fact that e operates at capacity, we obtain the following inequality, which proves the statement.

$$\begin{aligned} z_e(\theta') &= \int_0^{\theta'} f_e^+(\xi) - f_e^-(\xi + \tau_e)d\xi \\ &= \underbrace{z_e(\theta)}_{=\nu_e q_e(\theta)} + \int_\theta^{\theta'} f_e^+(\xi) - f_e^-(\xi + \tau_e)d\xi \\ &\geq \nu_e q_e(\theta) - \int_\theta^{\theta'} f_e^-(\xi + \tau_e)d\xi \\ &\geq \nu_e q_e(\theta) - \int_\theta^{\theta'} \nu_e d\xi \\ &= \nu_e(\theta + q_e(\theta) - \theta') > 0 \end{aligned}$$

Secondly, the entire flow in the queue at θ has left it exactly at time $\theta+q_e(\theta)$, as the following argumentation shows. The amount of flow leaving the queue between θ and $\theta + q_e(\theta)$ is equal to the amount of flow that leaves the edge between $\theta + \tau_e$ and $\theta + q_e(\theta) + \tau_e$. The insight that the queue is nonempty on W_e^θ yields the claim, as the following equality holds.

$$\int_\theta^{\theta+q_e(\theta)} \underbrace{f_e^-(\xi+\tau_e)}_{=\nu_e \text{ a.e.}} d\xi = q_e(\theta)\nu_e = z_e(\theta)$$

The assumption that queues operate at capacity guarantees that a flow particle which enters an edge e has to leave it at some point in the future. More precisely, a particle entering e at θ has to leave e by arriving at w exactly at time $\theta + q_e(\theta) + \tau_e$, that is the time of arrival θ, the queue delay $q_e(\theta)$ as well as the transit time τ_e. For notational convenience, we will refer to this deduced edge travel time by $T_e(\theta) := \theta + q_e(\theta) + \tau_e$. The important point to note here is that q_e and consequently T_e are both functions, which depend on a given flow over time f, despite the fact that it is not notationally comprised by these functions. We emphasize that the definition of T_e comprehends the travel duration through e subject to inflows into e.

2.3 Dynamic shortest paths and equilibria

Now that we have established a proper model in order to formulate the problem, we investigate the behaviour of lengths of s-t-paths, which are (game-theoretically) the strategies player θ may follow. As mentioned earlier the time a particle leaves an edge e, given that it entered e at time θ, is given by the time-dependent function $T_e(\theta) = \theta + \frac{z_e(\theta)}{\nu_e} + \tau_e$. It is not hard to see that T_e is locally absolutely continuous, as it is a sum of $AC_{loc}(\mathbb{R})$ functions (cp. Lemma A.2). In particular, T_e is differentiable almost everywhere.

Let us state two important consequences of this very insight.

Proposition 2.1
Let $e \in E$ be an edge. Then the following two statements hold.

i) $T_e'(\theta) \geq 0$ *for almost all* $\theta \in \mathbb{R}$

ii) $F_e^+(\theta) = F_e^-(T_e(\theta))$ *for all* $\theta \in \mathbb{R}$

Proof. Let θ be such that both $T_e'(\theta)$ exists as well as (4) is valid. Note that we have implicitly used that the union of two null-sets is a null-set, justifying that the two conditions are satisfied simultaneously almost everywhere. Simple differentiation leads to the following formula.

$$T_e'(\theta) = 1 + \frac{1}{\nu_e} z_e'(\theta) = 1 + \frac{1}{\nu_e}(f_e^+(\theta) - f_e^-(\theta+\tau_e))$$

It is now easy to see that the derivative is

$$T_e'(\theta) = \begin{cases} \frac{1}{\nu_e} f_e^+(\theta) & \text{if } z_e(\theta) > 0 \\ \max\left\{1, \frac{1}{\nu_e} f_e^+(\theta)\right\} & \text{otherwise,} \end{cases} \tag{7}$$

proving claim i). To prove the second claim, observe that

$$F_e^-(T_e(\theta)) = \int_0^{T_e(\theta)} f_e^-(\xi)d\xi = \int_0^{\theta+\tau_e} f_e^-(\xi)d\xi + \int_{\theta+\tau_e}^{(\theta+\tau_e)+q_e(\theta)} f_e^-(\xi)d\xi$$

and recall that $z_e(\zeta) > 0$ for all $\zeta \in W_e^\theta := [\theta, \theta+q_e(\theta)[$ and hence $f_e^-(\zeta+\tau_e) = \nu_e$ for almost all $\zeta \in W_e^\theta$.
We thus conclude the proof by simple computation.

$$\begin{aligned} F_e^-(T_e(\theta)) &= \int_0^{\theta+\tau_e} f_e^-(\xi)d\xi + \int_\theta^{\theta+q_e(\theta)} f_e^-(\xi+\tau_e)d\xi \\ &= \int_0^{\theta+\tau_e} f_e^-(\xi)d\xi + \int_\theta^{\theta+q_e(\theta)} \nu_e d\xi \\ &= F_e^-(\theta+\tau_e) + \nu_e q_e(\theta) \\ &= F_e^-(\theta+\tau_e) + z_e(\theta) \\ &= F_e^+(\theta) \end{aligned}$$

□

The above proposition yields two interesting characteristics of our modelling. First of all, we have $T_e \in AC_{loc}(\mathbb{R})$ and $T_e'(\theta) \geq 0$ almost everywhere in $\mathbb{R}$, hence T_e is nondecreasing (cp. Corollary A.1). We remind the reader that this reasoning requires stronger premises than mere continuity (a common counterexample is the well-known Cantor function). As $T_e(\theta)$ is the time a particle entering e at θ leaves e, we can interpret the property of this time being nondecreasing as follows: particles traveling through e and its queue respect the First-In-First-Out-principle (FIFO). In other words, consider two particles entering e at θ and θ' with $\theta' > \theta$. Particle θ' can by no means arrive earlier at the end of e than θ. Two particles cannot overtake each other within e and within its queue. This will be of importance for dynamic equilibria, which will be considered at the end of this section.
Furthermore, we have that $F_e^+(\theta) = F_e^-(T_e(\theta))$ for all θ. In simple terms, all the flow that enteres an edge e up to time θ has left e exactly at time $T_e(\theta)$. This is in conformity with our intuition regarding queues and shows that the modelling of queues is not flawed.

Our goal of this section is to properly define Nash flows over time or dynamic equilibria. As stated earlier, one can interpret time θ as a particle or player starting at s, wanting to travel to t as quickly as possible, that is chosing the shortest s-t-path. Given an s-v-path $P = (e_1, \ldots, e_j)$, we can now state the time needed to travel from s to v through P. Clearly, the particle will arrive at v at time

$$l^P(\theta) := T_{e_j} \circ \ldots \circ T_{e_1}(\theta).$$

Having a finite number of such (simple) s-v paths, we can easily define the earliest arrival time at v, starting at s at time θ, as follows.

$$l_v(\theta) := \min_{P \ s\text{-}v\text{-path}} l^P(\theta)$$

In this definition, cycles are of no importance, as we assumed that the sum of the transit times of a cycle is always positive. A rigorous justification can be found in the proof of the next lemma. The functions $(l_v(\cdot))_{v\in V}$ correspond to the shortest paths in G where the edge travel costs incorporate the queuing delays that occur while traveling from s towards a node v. These shortest paths are called *dynamic shortest paths.*
Given time θ we define $G_\theta = (V, E'_\theta)$ to be the *θ-shortest-path graph*, that is the subgraph of G containing all shortest paths at θ with respect to the earliest arrival time functions $(l_v(\theta))_{v\in V}$. For each edge $e = vw \in E$ the following equivalence holds:

$$e = vw \in E'_\theta \Leftrightarrow T_e(l_v(\theta)) = l_w(\theta). \tag{8}$$

Note that in general we have $T_e(l_v(\theta)) \geq l_w(\theta)$ for $e = vw \in E$.

We begin with a more or less apparent result regarding the dynamic shortest paths that proves to be useful in later arguments of this thesis. To not disrupt the flow of reading, we postpone the mostly technical proof to the appendix.

Lemma 2.1
Let $w \in V$, $\theta \in \mathbb{R}$. Then:

i) $l_w \in AC_{loc}(\mathbb{R})$ and $l_w(\cdot)$ is nondecreasing.

ii) $G_\theta = (V, E'_\theta)$ is acyclic and $l_w(\theta) = \begin{cases} \theta & \text{if } w = s \\ \min\limits_{e=vw\in\delta^-(w)} T_e(l_v(\theta)) & \text{if } w \neq s \end{cases}$.

We call an edge $e \in E_\theta$ *active* at θ, that is, e lies on a shortest path at time θ. By Θ_e we denote the set of times θ such that e is active at θ and by Θ_e^c we denote its complement.
Finally, we can define the dynamic equilibrium. In a Nash flow over time, every particle chooses the shortest s-t-path. Thus, flow is only sent over dynamic shortest paths.

Definition 2.5 (Dynamic Equilibrium)**.** A feasible flow over time f is called a dynamic equilibrium if and only if for each $e = vw \in E$ and almost all $\xi \in \mathbb{R}$ the following implication holds.

$$f_e^+(\xi) > 0 \Rightarrow \xi \in l_v(\Theta_e) \tag{9}$$

An equivalent though less graspable condition is that for almost all $\xi \in l_v(\Theta_e^c)$ it holds that $f_e^+(\xi) = 0$ (see [1] for a proof).

Remark 2.3. In [8] the dynamic equilibrium is defined slightly different. In fact, Koch and Skutella formulate a stronger criterion. Chapter 2.5 of [1] contains an demonstrative example of why the weaker definition as above is sufficient for the task, as the (weak) dynamic equilibrium is non-sensitive to modifications (of the inflow/outflow-rate functions) on null sets. Nonetheless, the intuition behind the dynamic equilibrium stays the same. In a dynamic equilibrium, flow is only (precisely: almost only, neglecting null sets) assigned to dynamic shortest paths.

Game-theoretically, almost every player or particle (or time) θ chooses a shortest s-t-path to travel along. As we have seen earlier, particles cannot overtake each other within edges or queues. Furthermore, $l_w(.)$ is nondecreasing for each $w \in V$. Hence, particles cannot be overtaken on their shortest s-t-path and the incurring cost for particle θ is exactly $l_t(\theta)$. A dynamic equilibrium is in fact a Nash flow over time.

2.4 Characterizing dynamic equilibria

When it comes to the computation of dynamic equilibria, the edges having a non-empty queue play an important role. In the upcoming sections, we, given a time θ, denote the set of all edges e whose queues have positive size at the earliest time a particle, starting from s at θ, could arrive at e as:

$$E^*_\theta = \{e = vw \in E \mid z_e(l_v(\theta)) > 0\}.$$

The following proposition by Cominetti et al. [1] will give us some deeper insight into the behaviour of active edges as well as edges that have a waiting queue, given a dynamic equilibrium.

Proposition 2.2
Let f be a dynamic equilibrium. Then we have that

i) $E^*_\theta \subseteq E'_\theta$,

ii) $E'_\theta = \{e = vw \in E \mid l_w(\theta) \geq l_v(\theta) + \tau_e\}$ *and*

iii) $E^*_\theta = \{e = vw \in E \mid l_w(\theta) > l_v(\theta) + \tau_e\}$.

Proof. Firstly, we prove that $E^*_\theta \subseteq E'_\theta$. In parts (ii) and (iii) we will simply prove the set equality by an inclusion-argument.

i) Let $e = vw \in E^*_\theta$, that is $z_e(l_v(\theta)) > 0$. But then, it must hold that $F^+_e(l_v(\theta)) > 0$, implying that, given dynamic equilibrium f, there exists a $\theta' \in [0, l_v(\theta)]$ such that $f^+_e(\theta') > 0$ and $\theta' \in l_v(\Theta_e)$. In other words, there must have existed a time $\theta' \leq \theta$ at which e has been active, that is e has been on some dynamic shortest s-t-path at time θ'. Let now $\theta' \leq \theta$ be the largest time at which e was active.
By construction, e is not active at time $\zeta \in]\theta', \theta]$. As f is a dynamic equilibrium, we have that $f^+_e(\xi) = 0$ for almost all $\xi = l_v(\zeta) \in l_v(]\theta', \theta])$. The queue must be nonempty over this interval, as $z_e(l_v(\theta)) > 0$. Precisely, for all $\xi \in]l_v(\theta'), l_v(\theta)]$ it holds that $z_e(\xi) > 0$. This is easy to see as if the queue was empty at some ξ, it would have stayed empty as no measurable amount of flow entered e up to $l_v(\theta)$. Equation 7 allows us to conclude that for almost all $\xi \in]l_v(\theta'), l_v(\theta)]$ it holds that $T'_e(\xi) = 0$. Thus T_e has to be constant on the interval $]l_v(\theta'), l_v(\theta)]$. By using the fact that e was active at θ' and by recalling that l_w is nondecreasing, we observe that

$$T_e(l_v(\theta)) = T_e(l_v(\theta')) = l_w(\theta') \leq l_w(\theta),$$

allowing the conclusion that $e \in E'_\theta$. This proves the claim.

ii) "$\subseteq$" Let $e \in E'_\theta$. Then $e \in \{e = vw \in E \mid l_w(\theta) \geq l_v(\theta) + \tau_e\}$, as the following inequality is valid.

$$l_w(\theta) = T_e(l_v(\theta)) = l_v(\theta) + \underbrace{q_e(l_v(\theta))}_{\geq 0} + \tau_e \geq l_v(\theta) + \tau_e$$

"$\supseteq$" Let $e = vw$ such that $l_w(\theta) \geq l_v(\theta) + \tau_e$. We distinguish the two cases whether $z_e(l_v(\theta)) = 0$ or $z_e(l_v(\theta)) > 0$. In the first case, the claim is obvious, as

$$l_w(\theta) \geq l_v(\theta) + \tau_e = l_v(\theta) + \frac{z_e(l_v(\theta))}{\nu_e} + \tau_e = T_e(l_v(\theta))$$

and thus $e \in E'_\theta$. If on the other hand $z_e(l_v(\theta)) > 0$, we immediately obtain that $e \in E^*_\theta \subseteq E'_\theta$ using (i), which proves the claim.

iii) "$\subseteq$" Let $e \in E^*_\theta$, that is $z_e(l_v(\theta)) > 0$. By (i) we have that $e \in E'_\theta$, which is equivalent to the validity of the equation $l_w(\theta) = T_e(l_v(\theta))$. But then we have demonstrated the validity of the first inclusion, as the following inequality follows immediately:

$$l_w(\theta) = T_e(l_v(\theta)) = l_v(\theta) + \underbrace{\frac{z_e(l_v(\theta))}{\nu_e}}_{>0} + \tau_e > l_v(\theta) + \tau_e$$

"$\supseteq$" Let $e = vw$ such that $l_w(\theta) > l_v(\theta) + \tau_e$. By the definition of l_w we have that

$$l_w(\theta) \leq T_e(l_v(\theta)) = l_v(\theta) + \frac{z_e(l_v(\theta))}{\nu_e} + \tau_e.$$

Combining these two inequalities, we obtain

$$\begin{aligned} l_v(\theta) + \tau_e &< l_v(\theta) + \frac{z_e(l_v(\theta))}{\nu_e} + \tau_e \\ \Leftrightarrow 0 &< \frac{z_e(l_v(\theta))}{\nu_e} \end{aligned}$$

which proves the statement.

□

Interestingly, (i) is somewhat counterintuitive. While one would expect that the queue of some edge could grow large enough (and consequently its queue delay) to leave the dynamic shortest path network, this is never the case in a dynamic equilibrium. An edge having a non-empty queue lies on a shortest s-w-path for some w. The intuition behind this is the following. If the queue of some edge had grown large enough for the edge to be not on a shortest path anymore, then the last particles in the queue would have chosen the edge even though it was not on a shortest path. This contradicts the intuition behind the dynamic equilibrium.

In order to further analyze dynamic equilibria and to obtain an idea on how to compute one, we state the following characterization.

Theorem 2.1
Let f be a feasible flow over time. The following statements are equivalent.

i) f is a dynamic equilibrium.

ii) For each $e = vw$ and all θ we have $F_e^+(l_v(\theta)) = F_e^-(l_w(\theta))$.

iii) For each $e = vw$ and almost all θ we have

$$e \notin E'_\theta \Rightarrow f_e^+(l_v(\theta))l'_v(\theta) = 0.$$

Proof. We begin by proving the equivalence $(i) \Leftrightarrow (ii)$. Therefore, let θ' and I_θ as in Lemma A.4. Using the definition of θ' and Proposition 2.1, we obtain

$$\begin{aligned} F_e^+(l_v(\theta)) - F_e^-(l_w(\theta)) &= F_e^+(l_v(\theta)) - F_e^-(T_e(l_v(\theta')) \\ &= F_e^+(l_v(\theta)) - F_e^+(l_v(\theta')) \\ &= \int_{l_v(\theta')}^{l_v(\theta)} \underbrace{f_e^+(\xi)}_{\geq 0} d\xi \geq 0 \end{aligned} \tag{10}$$

where technically, the non-negativity, follows from the monotonicity of l_v and the fact that $\theta' \leq \theta$.
To see why the following statements are equivalent, recall the fact from Lemma A.4 that $\Theta_e^c = \bigcup_{\theta''} I_{\theta''}$ and Lemma A.3.

$$\begin{aligned} &F_e^+(l_v(\theta)) - F_e^-(l_w(\theta)) = 0 \text{ for all } \theta \\ &\Leftrightarrow f_e^+ \text{ vanishes a.e. on }]l_v(\theta'), l_v(\theta)] = l_v(I_\theta) \text{ for all } \theta \\ &\Leftrightarrow f_e^+ \text{ vanishes a.e. on } \bigcup_{\theta''} l_v(I_{\theta''}) = l_v(\Theta_e^c) \end{aligned}$$

Thus if Equation 10 holds with equality for all θ and e, then we clearly have that f is a dynamic equilibrium and vice versa.
In order to prove the equivalence of (ii) and (iii), observe that we can rewrite Equation 10 using the change of variable formula from Corollary A.3.

$$\begin{aligned} F_e^+(l_v(\theta)) - F_e^-(l_w(\theta)) &= \int_{l_v(\theta')}^{l_v(\theta)} f_e^+(\xi)d\xi \\ &= \int_{\theta'}^{\theta} f_e^+(l_v(t))l'_v(t)dt \geq 0 \end{aligned}$$

Similarly to the logic above, we have that $F_e^+(l_v(\theta)) - F_e^-(l_w(\theta)) = 0$, if and only if $f_e^+(l_v(t))l'_v(t) = 0$ for almost all $t \in I_\theta$. Hence (ii) holds if and only if $f_e^+(l_v(t))l'_v(t) = 0$ for almost all $t \in \bigcup_{\theta''} I_{\theta''} = \Theta_e^c$, which proves that $(ii) \Leftrightarrow (iii)$. □

2.5 The constitution of dynamic equilibria

Now that we have found a way to characterize dynamic equilibria, we have got a deeper insight into their essence. This is going to help us understand what constitutes a Nash flow over time and in particular this yields the theoretical foundation allowing their computation. In the first part of this section, we introduce the concept of normalized thin flows with resetting, which is strongly linked to the constitution of Nash flows. This link will be explained thoroughly in the second part.

2.5.1 Normalized thin flows with resetting

We introduce normalized thin flows with resetting in a more general setting. Let therefore $u_0 \geq 0$ and let $G' = (V, E')$ be a subgraph of G, where $E' \subseteq E$ is acyclic and having the property that for all $v \in V$ there is an s-v-path in E'. Furthermore, let $E^* \subseteq E'$. Whenever we mention E^* or E', we refer to a pair (E^*, E') of edge sets satisfying these conditions. Before we can proceed with the proper definition, one more notation has to be explained.
In the following we denote by

$$K(E', u_0)$$

the set of all static s-t-flows $(x'_e)_{e \in E} \geq 0$ of value $u_0 \geq 0$ such that $x'_e = 0$ for $e \in E \setminus E'$. With each $x' \in K(E', u_0)$ we associate the labels $l' \in \mathbb{R}^{|V|}$ given by

$$\begin{aligned} l'_s &= 1 \\ l'_w &= \min_{e=vw \in E'} \rho_e(l'_v, x'_e) \text{ for } w \in V \setminus \{s\} \end{aligned}$$

where we define

$$\rho_e(l'_v, x'_e) := \begin{cases} \frac{x'_e}{\nu_e} & \text{if } e \in E^* \\ \max\left\{l'_v, \frac{x'_e}{\nu_e}\right\} & \text{if } e \notin E^*. \end{cases}$$

At this point, these labels might be notationally confusing, as we used l'_v earlier in this thesis as the derivative of the earliest arrival time function l_v. We emphasize that l'_v is now a real value, instead of a function. The connection between the value l'_v and the function l'_v will be explained shortly. Note that l'_w is properly defined, as E' is acyclic.

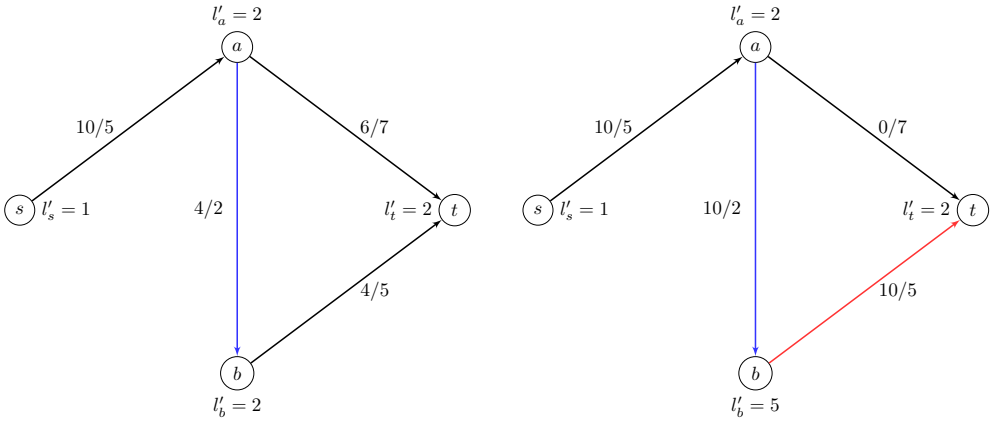
Definition 2.6. $x' \in K(E', u_0)$ is called a normalized thin flow (NTF) of value u_0 with resetting on $E^* \subseteq E'$ if for every edge $e = vw \in E'$ the following implication is valid.

$$l'_w < \rho_e(l'_v, x'_e) \Rightarrow x'_e = 0$$

Remark 2.4. The function $x' \mapsto l'$ assigns a unique set of node labels to each $x' \in K(E', u_0)$. Therefore, we often call the tuple (x', l') an NTF, although technically speaking x' is the actual normalized thin flow.

Example 2.1. Figure 2.2 displays a network and two different s-t-flows. Out of four edges, only the edge ab (highlighted in blue) is a resetting edge.
While the flow in Figure 2.2a is an NTF, the flow in Figure 2.2b is not. The definition of NTFs is violated on edge $e = bt$, which is colored in red. More precisely, we have that $l'_t = 2 < 5 = \max\left\{l'_b, \frac{x'_e}{\nu_e}\right\} = \rho_e(l'_b, x'_e)$ but $x'_e = 10 > 0$.

Remark 2.5. Normalized thin flows are special (static) s-t-flows. The authors of [8] were the first to introduce the concept of thin flows, which are closely related to **normalized** thin flows. They further gave an interpretation of the rather abstract concept. Suppose for a moment that $E^* = \emptyset$, that



(a) A normalized thin flow. (b) A flow, violating the NTF condition.

Figure 2.2: A network with resetting on $E^* := \{ab\}$ and inflowrate $u_0 = 10$. Next to each node v the label l'_v of the respective flow is indicated. Furthermore, the label next to an edge e indicates the ratio of flow on e to the service rate of e.

is there are no resetting edges. For each path P to some node w the congestion of P could be defined as the maximum congestion over all edges of P. Precisely, define the congestion of P as $\max_{e \in P} x'_e/\nu_e$. Then l'_w is exactly the congestion of all flow carrying s-w-paths and furthermore a lower bound on the congestion of all s-w-paths. However, if there is some edge $e \in E^* \cap P$, then the congestion on this path is "resetted" to x'_e/ν_e. The congestions of all edges on P before e are forgotten, as they put it. The concept of **normalized** thin flows is due to the authors of [1], who proposed a modification to ensure some kind of uniqueness (see [3]).

We conclude this subsection with two results, proving the existence of NTFs as well as the uniqueness of their labels.

Theorem 2.2
Let $u_0 \geq 0$ and (E^*, E') as above. Then there is an NTF of value u_0 with resetting on $E^* \subseteq E'$.

Proof. We make use of a fixed point argument. Denote therefore $K := K(E', u_0)$ and define the set-valued map $\Gamma : K \to 2^K$ as follows:

$$x' \mapsto \{y' \in K \mid y'_e = 0 \text{ for all } e = vw \in E' \text{ with } l'_w < \rho_e(l'_v, x'_e)\}$$

where l' are the unique labels associated with the flow x'. Observe that a fixed point $x^* \in \Gamma(x^*)$ is an NTF and thus the existence of such a fixed point implies the claim. In the following, we treat flows as vectors in $\mathbb{R}^{|E|}$ and similarly labels as vectors in $\mathbb{R}^{|V|}$.
We begin by proving that K is compact convex. Let $x', x'' \in K$ with labels l', l''. Let $\lambda \in [0, 1]$ and $z := \lambda x' + (1 - \lambda)x''$. Clearly $z_e \geq 0$ for all e and $z_e = 0$ if $e \notin E'$. It remains to show that the flow conservation still holds. Clearly, for $v \in V \setminus \{t\}$ we have

$$\sum_{e \in \delta^+(v)} z_e - \sum_{e \in \delta^-(v)} z_e = \begin{cases} u_0 & \text{if } v = s \\ 0 & \text{if } v \in V \setminus \{s, t\} \end{cases}$$

and thus $z \in K$, proving that K is convex. Clearly, K is bounded as for each $x' \in K$, $0 \leq x' \leq u_0 \cdot \mathbb{1}$ entry-wise. Furthermore, K is closed, as for every convergent sequence (x^n) in K with limit x^* we have that $x^* \in K$, which one could justify as follows. Undoubtedly we have $x^*_e = 0$ for $e \notin E'$. Furthermore, for $v \in V \setminus \{t\}$

$$\sum_{e \in \delta^+(v)} x^*_e - \sum_{e \in \delta^-(v)} x^*_e = \lim_{n \to \infty} \left(\sum_{e \in \delta^+(v)} x^n_e - \sum_{e \in \delta^-(v)} x^n_e \right) = \begin{cases} u_0 & \text{if } v = s \\ 0 & \text{if } v \in V \setminus \{s,t\} \end{cases}$$

holds. Hence, K is compact convex. Obviously, K is non-empty, as there is an s-t-path in E'.

We continue by showing that for x' the set $\Gamma(x')$ is non-empty, which is easy to see. Given x' and labels l', we have that $l'_w = \rho_e(l'_v, x'_e)$ for some $e = vw$ for each $w \in V \setminus \{s\}$. Thus there exists an s-t-path such that no edge $e = vw$ has the property $l'_w < \rho_e(l'_v, x'_e)$. Let $y' \in K$ be the flow that sends u_0 along this path. Then $y' \in \Gamma(x')$. We omit the proof of the convexity of $\Gamma(x')$, as it works analogously to the proof of the convexity of K.
To be capable of applying Kakutanis Fixed Point Theorem (see Theorem A.3) it remains to show that the function graph $\{(x, y) \mid y \in \Gamma(x)\}$ is closed. Let therefore $(x^n), (y^n)$ be two sequences in K with limt point x^* and y^* such that $y^n \in \Gamma(x^n)$. K is compact and thus $x^*, y^* \in K$. Let l^n be the associated labels of x^n and similarly l^* the labels of x^*. We show that $y^* \in \Gamma(x^*)$ or equivalently that $y^*_e = 0$ for all $e = vw \in E'$ with $l^*_w < \rho_e(l^*_v, x^*_e)$. Suppose $y^*_e > 0$ and $l^*_w < \rho_e(l^*_v, x^*_e)$. As $y^n \to y^*$ there must exist $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, y^n_e is also positive. But $y^n_e \in \Gamma(x^n)$ implies that for all $n \geq n_0$ we have $l^n_w = \rho_e(l^n_v, x^n_e)$. Regardless of $e \in E^*$ or not, the function $\rho_e(l^n_v, x^n_e)$ is continuous. Additionally, the map $x' \to l'$ associating labels to a flow $x' \in K$ is continuous, as it is just a composition of continuous functions. The continuity yields a contradiction as the following equality holds:

$$l^*_w = \lim_{n \to \infty} l^n_w = \lim_{n \to \infty} \rho_e(l^n_v, x^n_e) = \rho_e(l^*_v, x^*_e).$$

Kakutanis fixed point theorem yields the desired result.

□

Lemma 2.2
Let $u_0 \geq 0$ and (E^*, E') as above. Let $(x', l'), (y', g')$ be two NTFs of value u_0 with resetting on $E^* \subseteq E'$. Then the labels are the same, that is for all $w \in V$ we have $l'_w = g'_w$.

For a proof see [1].

2.5.2 Underlying static flows

By Theorem 2.1 (ii) a dynamic equilibrium f has the property that for each edge $e = vw$ and time θ we have that $F^+_e(l_v(\theta)) = F^-_e(l_w(\theta))$. This

motivates the definition of yet another class of static flows, namely *underlying static flows*. Given a dynamic equilibrium f and time θ we refer to $x(\theta) := (x_e(\theta))_{e\in E}$ as the *underlying static flow* at time θ, where $x_e(\theta) := F_e^+(l_v(\theta)) = F_e^-(l_w(\theta))$ for edge $e = vw \in E$.
The following short lemma justifies the denotation of underlying static flows as actual flows.

Lemma 2.3
Let $\theta \in \mathbb{R}$. Then:

i) $x(\theta)$ is a static s-t-flow of value $U(\theta)$

ii) For $e \in E$, x_e is differentiable almost everywhere and for almost all θ we obtain that $x'(\theta) := (x'_e(\theta))_{e\in E}$ is a static s-t-flow of value $u(\theta)$ with the property that $x'_e(\theta) = 0$ for $e \notin E'_\theta$.

Proof. Recall the flow over time conservation constraint (Equation 2). For an edge e we have $f_e^+, f_e^- \in L^1_{loc}(\mathbb{R})$ and we thus prove the first claim by mere computation. Let $v \in V \setminus \{t\}$. Integration of the flow conservation constraint over $[0, l_v(\theta)]$ yields:

$$\begin{aligned}
&\sum_{e\in\delta^+(v)} x_e(\theta) - \sum_{e\in\delta^-(v)} x_e(\theta) \qquad (11)\\
&= \sum_{e\in\delta^+(v)} F_e^+(l_v(\theta)) - \sum_{e\in\delta^-(v)} F_e^-(l_v(\theta))\\
&= \sum_{e\in\delta^+(v)} \int_0^{l_v(\theta)} f_e^+(\xi)d\xi - \sum_{e\in\delta^-(v)} \int_0^{l_v(\theta)} f_e^-(\xi)d\xi\\
&= \int_0^{l_v(\theta)} \sum_{e\in\delta^+(v)} f_e^+(\xi) - \sum_{e\in\delta^-(v)} f_e^-(\xi)d\xi\\
&= \begin{cases} \int_0^{l_s(\theta)} u(\xi)d\xi = U(\theta) & \text{if } v = s\\ 0 & \text{if } v \in V\setminus\{s,t\}. \end{cases}
\end{aligned}$$

Note that in the last equality we used the fact that $l_s(\theta) = \theta$. Hence, $x(\theta)$ satisfies the flow conservation constraints. Furthermore, $x_e(\theta) \geq 0$ for all e. As we do not have any capacity constraints, this concludes the proof of the first claim.
Regarding the second claim, observe that x_e is locally absolutely continuous by Lemma A.2. Hence, x_e is differentiable almost everywhere. As we have a finite number of different edges it is not hard to see that for almost all θ the derivatives $x'_e(\theta)$ exist simultaneously for all $e \in E$. Differentiation of Equation 11 then yields the validity of the claim, as

$$\sum_{e\in\delta^+(v)} x'_e(\theta) - \sum_{e\in\delta^-(v)} x'_e(\theta) = \begin{cases} U'(\theta) = u(\theta) & \text{if } v = s\\ 0 & \text{if } v \in V\setminus\{s,t\} \end{cases}.$$

Clearly, for almost all $\theta \in \mathbb{R}$, $e \notin E'_\theta$ implies

$$x'_e(\theta) = f_e^+(l_v(\theta))l'_v(\theta) = 0$$

by Theorem 2.1. □

It becomes more and more apparent how one could develop an approach to calculate Nash flows over time. Theorem 2.1 gives us an interesting characterization of dynamic equilibria. If we knew the label functions l_w and the underlying static flow x, then it would be easy to compute the inflow/outflow functions of each edge in order to get the dynamic equilibrium. Both l_w and x_e are locally absolutely continuous, hence differentiable almost everywhere. The idea that we outline in the next paragraphs and sections is the approach of finding the derivatives l'_w and x'_e in order to obtain l_w and x_e by simple integration.

The purpose of the introduction of NTFs is established in the next theorem.

Theorem 2.3
Let f be a dynamic equilibrium and time $\theta \in \mathbb{R}$ such that the right derivatives $u_0 := \frac{dU}{d\theta^+}(\theta), l'_v := \frac{dl_v}{d\theta^+}(\theta)$ and $x'_e := \frac{dx_e}{d\theta^+}(\theta)$ exist for $v \in V, e \in E$. Then $x' = (x'_e)_{e\in E} \in K(E'_\theta, u_0)$ is an NTF of value u_0 with resetting on $E^*_\theta \subseteq E'_\theta$ and corresponding labels $l' = (l'_v)_{v\in V}$.

Proof. First, it is necessary to show that $x' \in K(E'_\theta, u_0)$. This however follows immediately from Lemma 2.3 (ii). It remains to show that l' are the labels associated with x' and that (x', l') is an NTF.

Clearly, we have $l'_s = 1$. We prove that $l'_w = \min_{e=vw\in E'_\theta} \rho_e(l'_v, x'_e)$ for all $w \in V \setminus \{s\}$ by distinguishing between two subset of E'_θ, namely:

- $E^*_+ := \{e = vw \in E'_\theta \mid \exists \epsilon > 0 \ \forall \xi \in]l_v(\theta), l_v(\theta) + \epsilon[: z_e(\xi) > 0\}$, which is the set of all edges e that have a queue at $l_v(\theta)$ or are about to build one up, and
- E'_+ as the set of edges $e = vw$ without queue at $l_v(\theta)$ but active along a strictly decreasing sequence $\theta_n \downarrow \theta$.

Note that due to the continuity of z_e we have that $E^*_\theta \subseteq E^*_+$. Let $e = vw \in E'_\theta$. We prove four different claims.

i) Claim: $l'_w \geq l'_v$ for $e \in E'_+$
Let $e \in E'_+$. e is active at every $\theta_n, n \in \mathbb{N}$ and also at θ. We obtain:

$$l_w(\theta_n) = T_e(l_v(\theta_n)) = l_v(\theta_n) + q_e(l_v(\theta_n)) + \tau_e \geq l_v(\theta_n) + \tau_e$$
$$l_w(\theta) = T_e(l_v(\theta)) = l_v(\theta) + q_e(l_v(\theta)) + \tau_e = l_v(\theta) + \tau_e$$

Subtraction of these equations from another yields

$$l_w(\theta_n) - l_w(\theta) \geq l_v(\theta_n) - l_v(\theta).$$

Dividing both sides by $\theta_n - \theta > 0$ yields $l'_w \geq l'_v$ for $n \to \infty$.

ii) Claim: $l'_w \leq l'_v$ for $e \in E'_\theta \setminus E^*_+$
Let $\theta_n \downarrow \theta$ with $z_e(l_v(\theta_n)) = 0$. Then we have

$$l_w(\theta_n) \leq T_e(l_v(\theta_n)) = l_v(\theta_n) + \tau_e$$

which implies

$$l_w(\theta_n) - l_w(\theta) \leq l_v(\theta_n) - l_v(\theta)$$

and thus also the claim, using a similar argument as in (i).

iii) Claim: $l'_w \geq x'_e/\nu_e$ for $e \in E'_\theta$
Observe that, using the fact that $f_e^-(\xi) \leq \nu_e$ a.e., for $\theta' > \theta$ the following inequality holds.

$$\frac{1}{\nu_e}(x_e(\theta') - x_e(\theta)) = \frac{1}{\nu_e} \int_{l_w(\theta)}^{l_w(\theta')} f_e^-(\xi) d\xi \leq l_w(\theta') - l_w(\theta) \qquad (12)$$

Again, dividing by $\theta' - \theta$ and considering the limit $\theta' \to \theta$ yields the statement.

iv) Claim: $l'_w = x'_e/\nu_e$ for $e \in E^*_+$
By definition of E^*_+ we have $z_e(l_v(\theta')) > 0$ for $\theta' > \theta$ close to θ. As observed in section 2 the queue remains nonempty on the interval $W_e^{l_v(\theta')} = [l_v(\theta'), l_v(\theta') + q_e(l_v(\theta'))[$. As queues operate at capacity, this implies that $f_e^-(\xi + \tau_e) = \nu_e$ for almost all $\xi \in W_e^{l_v(\theta')}$ or equivalently $f_e^-(\xi) = \nu_e$ for almost all $\xi \in [l_v(\theta') + \tau_e, l_w(\theta')[$, where we used that $l_v(\theta') + q_e(l_v(\theta')) + \tau_e = l_w(\theta')$. Then we have that $f_e^-(\xi) = \nu_e$ a.e. for a sufficiently small interval to the right of $l_w(\theta)$. For close enough θ', Equation 12 holds with equality, which proves the claim using the exact same argument as before.

Let $w \in V \setminus \{s\}$. Using (ii) and (iv) we have $l'_w \leq \min_{e=vw \in E'_\theta} \rho_e(l'_v, x'_e)$ with equality if $E^*_\theta \cap \delta^-(w) \neq \emptyset$. If the latter set is empty, the equality holds nonetheless. In that case let $\theta_n \downarrow \theta$ be a sequence and $e_n \in E'_{\theta_n}$ a sequence of active edges. Take now a subsequence with $e_n = vw$ constant. By definition then $e = vw \in E'_+$. (i) and (iii) yield $l'_w \geq \rho_e(l'_v, x'_e)$, proving the equality. Thus, l' are the unique labels associated with x'. It remains to show that x' is an NTF. Suppose there exists $e = vw$ such that $x'_e > 0$ and $l'_w < \rho_e(l'_v, x'_e)$. Then using (iv) we know that $e \notin E^*_\theta$. $x'_e > 0$ implies $x_e(\theta') > x_e(\theta)$ for $\theta' > \theta$ close to θ. Thus e is active on a sequence $\theta_n \downarrow \theta$ and $e \in E'_+$. But then, (i) and (iii) yields a contradiction, as $l'_w \geq \rho_e(l'_v, x'_e)$. This concludes the proof. □

In other words, given a dynamic equilibrium, the derivatives of the earliest arrival time functions and of the underlying static flows with respect to some time θ, form a normalized thin flow with resetting on E^*_θ. Again, knowing that U, l_v for $v \in V$ and x_e for $e \in E$ are differentiable almost everywhere, it is quite evident that these (countably many) functions are simultaneously differentiable almost everywhere. Although no proof has been given, the authors of [8] claim that the reverse direction of Theorem 2.3 holds as well, that is if for almost all times θ the derivatives of the earliest arrival time functions and underlying static flows of a feasible flow over time are normalized thin flows, then the dynamic flow is a dynamic equilibrium. Our particular interest however lies in a corollary of the reverse direction, to which we pay attention in the next section.

3 Computation of Nash flows over time

In the last section we have seen how to characterize Nash flows over time. Furthermore, we obtained an important result, namely that the derivatives of the underlying static flow functions x_e as well as earliest arrival time function l_v have a specific form: they constitute a thin flow with resetting. Koch and Skutella [8] used this insight to prove the correctness of a method to compute a Nash flow over time in the case of a constant inflow function $u \equiv u_0 \in \mathbb{R}_{\geq 0}$. The setting is as follows. Let $\theta_k \in \mathbb{R}$ and f be a feasible flow over time that satisfies $F_e^+(l_v(\theta)) = F_e^-(l_w(\theta))$ for all edges $e = vw$ and times $\theta \leq \theta_k$. Informally speaking, f is a Nash flow over time up to time θ_k (with respect to constant inflow u_0). Such flows are called *restricted Nash flows over time* on $]-\infty, \theta_k[$. Koch and Skutella formally introduce this definition by stating that f is such a restricted Nash flow if it is a Nash flow over time with respect to the inflow function

$$u(\theta) = \begin{cases} u_0 & \text{if } \theta < \theta_k \\ 0 & \text{if } \theta \geq \theta_k \end{cases} .$$

They describe the α-extension method, allowing the extension of f, which is a procedure to find $\theta_{k+1} = \theta_k + \alpha_k > \theta_k$ such that the extension is a restricted Nash flow over time on $]-\infty, \theta_{k+1}[$.

Extending f works as follows:

1. Find an NTF (x', l') of value u_0 with resetting on $E^*_{\theta_k} \subseteq E'_{\theta_k}$.

2. Compute $\theta_{k+1} = \theta_k + \alpha_k$ where $\alpha_k > 0$ the largest value satisfying:

$$l_w(\theta_k) - l_v(\theta_k) + \alpha_k(l'_w - l'_v) \geq \tau_e \text{ for all } e = vw \in E^*_{\theta_k} \tag{13}$$
$$l_w(\theta_k) - l_v(\theta_k) + \alpha_k(l'_w - l'_v) \leq \tau_e \text{ for all } e = vw \in E \setminus E'_{\theta_k}. \tag{14}$$

3. Extend l_v and f_e^+, f_e^- as follows.

$$l_v(\theta) = l_v(\theta_k) + (\theta - \theta_k)l'_v \text{ for } v \in V \text{ and } \theta \in [\theta_k, \theta_{k+1}]$$
$$f_e^+(\xi) = x'_e/l'_v \text{ for } e = vw \in E \text{ and } \xi \in [l_v(\theta_k), l_v(\theta_{k+1})[$$
$$f_e^-(\xi) = x'_e/l'_w \text{ for } e = vw \in E \text{ and } \xi \in [l_w(\theta_k), l_w(\theta_{k+1})[$$

Condition (13) can be interpreted as that θ_{k+1} may not be larger than the maximum time up to which the queue of edge $e \in E^*_{\theta_k}$ does not vanish. (14) enforces θ_{k+1} to be not larger than the maximum time up to which edges that are not in the θ_k-shortest-path network remain inactive. Hence α_k is chosen as the maximum range on which inactive edges remain inactive and non-empty queues remain non-empty.

We argue why each step of the α-extension method is properly defined. First of all, Theorem 2.2 implies that x' exists. The corresponding labels l' are unique as proved in Lemma 2.2. At this point, it remains to be seen whether an NTF is computable in finite time and more precisely, how to do it. In the

next section, we will more closely investigate NTFs and how to find them. Secondly, we give a reasoning why α_k can be chosen to be of positive value. Note that α_k is not necessarily a real number, but could also become infinitely large.

- Let $e = vw \in E^*_{\theta_k}$. By Proposition 2.2 we have that $l_w(\theta_k) - l_v(\theta_k) > \tau_e$. Thus there exists $\alpha' > 0$ such that Equation 13 is valid for all $0 < \alpha_k \leq \alpha'$.

- Let $e = vw \in E \setminus E'_{\theta_k}$. Similarly, Proposition 2.2 yields $l_w(\theta_k) - l_v(\theta_k) < \tau_e$, thus there exists $\alpha'' > 0$ such that Equation 14 holds for all $0 < \alpha_k \leq \alpha''$.

Combining these two insights, it is not hard to see that a positive α_k satisfying the constraints simultaneously must exist. To justify the third step in the above method, we argue that the division is well-defined. If $l'_v = 0$ where $v \in e$ for some $e \in E$, then we have $l_v(\theta_{k+1}) = l_v(\theta_k)$ and thus the interval on which the inflow/outflow functions can be extended is empty, that is they do not need to be extended.
The following theorem justifies the correctness of the above method. We omit the proof and refer to [8].

Theorem 3.1
Let f be a restricted Nash flow over time on $]-\infty, \theta_k[$. Then the α-extension of f is a restricted Nash flow over time on $]-\infty, \theta_{k+1}[$.

Remark 3.1. As mentioned in [1] without further explanation, this approach can easily be generalized to piecewise constant inflow rate functions. Although we give no proof whatsoever, an intuitive approach would be to constrain α_k further by enforcing that the inflow rate does not change on an interval $[\theta_k, \theta_k + \alpha_k[$, that is $u(\xi) = u(\theta_k)$ for $\xi \in [\theta_k, \theta_k + \alpha_k[$. A positive α_k can be chosen nonetheless, given that u is piecewise constant.

This theorem yields two interesting results. Firstly, we can extend dynamic equilibria (in the strong sense of Koch and Skutella [8], cp. [1, Chapter 2.5]) by a positive margin using this approach, where we assume that NTFs can be found in finite time. If the number of such α-extension intervals is bounded, we can compute the entire dynamic equilibrium by iterating this method. To the best of our knowledge, there are at this moment in time no results regarding the questions whether the number of intervals can always be bounded. Recently, Cominetti, Correa and Olver analyzed the long term behaviour of dynamic equilibria in [2], given a constant inflow rate u_0. They proved that a dynamic equilibrium in a network, satisfying the requirement that the constant inflow u_0 is not larger than the minimal queuing capacity among all s-t-cuts, that is

$$u_0 \leq \min_{C\, s\text{-}t\text{-Cut}} \sum_{e \in C} \nu_e,$$

reaches a so-called *steady state* in finite time. A *steady state* is a point in time from which on queue lengths remain constant. While this is certainly an interesting result, it has no implication regarding whether such networks

have finitely many intervals, as it is still an open question whether it could happen that "queues alternate infinitely often over a finite time interval", as they put it. Furthermore, there certainly exist graphs where the dynamic equilibrium never reaches a steady state but where the number of intervals is finite (consider e.g. the single-edge network connecting just the source to the sink with lower service rate than inflow). It is worth noting that in the α-extension method, the composition of such intervals is deterministic, regardless of which NTFs are chosen within the iterations of the method. Clearly, the chosen α_k is only dependent on the labels l' in (13) and (14). However, the labels do, by virtue of Lemma 2.2, not depend on the NTF x'.

Secondly, the correctness of this method can be used to prove the existence of dynamic equilibria with respect to constant inflow rates (see [1], [3]). To this point, there have however been proofs showing the existence of dynamic equilibria for a much larger family of inflow rate functions, that is $u \in L^p(\mathbb{R}_{\geq 0}) := \left\{ f \mid f \text{ measurable}, \int_0^\infty |f(t)|^p dt < \infty \right\}$ for $1 < p < \infty$.

4 Computation of normalized thin flows

As we have seen in the previous section, the computation of a dynamic equilibrium basically boils down to the problem of finding normalized thin flows. Recall that the problem can be summarized as follows: Given $u_0 \geq 0$ and a pair of edge sets (E^*, E') such that $E^* \subseteq E' \subseteq E$ with E' acyclic and for all $v \in V$ there is an s-v-path in E', find an NTF $x' \in K(E', u_0)$ with resetting on E^*.
At this point in time, it is not clear whether the problem is actually solvable efficiently, apart from the special case that the set of resetting edges is empty, that is $E^* = \emptyset$. For that particular case, Koch and Skutella provided a polynomial time algorithm to compute a thin flow in [8]. However, finding an NTF given an instance containing resetting edges seems to be a much harder task. We discuss an existing approach and try to give improvements.

4.1 An existing approach

Cominetti, Correa and Larré [1] propose a "finite (exponential time) algorithm" to compute NTFs. In this section we take a closer look at this algorithm.

In order to compute an NTF, the suggested algorithm first "guesses" the set $E'_0 := \{e = vw \in E' \mid l'_w = \rho_e(l'_v, x'_e)\}$, that is the set of edges which potentially could send flow, given some NTF x' with labels l'. By guessing we intend to say that the algorithm takes advantage of a blackbox, taking (E^*, E') and u_0 as input and returning the set E'_0. In reality, we have to iterate over all possible $E'_0 \subseteq E'$. Hence, in the following, we treat each possible candidate $E'_0 \subseteq E'$ as a "guess". We say that $E'_0 \subseteq E'$ has been guessed correctly if there exists an NTF (x', l') of value u_0 with resetting on E^* such that

$$E'_0 = \{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\},$$

that is the two sets coincide.

Remark 4.1. While there might be different NTFs, the correct guess set E'_0 is unique. Consider two different NTFs x' and x'' with labels l' and l''. Clearly, the labels are the same, that is $l' = l''$. Suppose now that the sets E'_0 and E''_0 differ. Without loss of generality, there exists some edge $e = vw$ such that $l'_w < \rho_e(l'_v, x'_e)$ but $l'_w = \rho_e(l'_v, x''_e)$. Combining these two equations, we have $\rho_e(l'_v, x''_e) < \rho_e(l'_v, x'_e)$. Note that by the definition of NTFs $x'_e = 0$. On the one hand, if $e \in E^*$, then this implies $\rho_e(l'_v, x''_e) < 0$. On the other hand, if $e \notin E^*$, then we have that $\max\{l'_v, x''_e/\nu_e\} < \max\{l'_v, 0\} = l'_v$. Both cases contradict the assumption that E'_0 and E''_0 differ. We thus often refer to E'_0 without explicitly stating the NTF (x', l').

A naive guessing routine, that is simple iteration over all subsets of E', could lead to up to $2^{|E'|}$ many iterations in the worst case. For each guess of E'_0 the algorithm solves the following optimization problem, which can be reformulated as a mixed integer linear program (to which we refer to by

(P)).

$$\max_{(x',l')} \left\{ \sum_{w \in V} l'_w \mid x' \in K(E'_0, u_0);\ l'_s = 1;\ l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e) \text{ for } w \in V \setminus s \right\}$$

We emphasize that (P) changes with a different guess of E'_0, although it is not captured by the notation.
In the paper, it remains unclear whether the constraint $l'_w = \rho_e(l'_v, x'_e)$ for $e = vw \in E'_0$ should be part of the optimization problem. Although the authors did not explicitly list it in the set of constraints, the guessing procedure suggests to take advantage of the additional information. In order to dissolve this ambiguity, we will consider both cases, that is leaving aforementioned constraint out or adding it. First we make an observation, which is crucial for the second possible interpretation of their algorithm, that is the version in which the additional constraints are part of the optimization problem.

Observation 4.1. Let E'_0 be guessed correctly with NTF (x', l'). For $w \in V \setminus \{s\}$, the definition of NTFs yields $l'_w = \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$ and thus there exists $e = vw \in E'$ such that $l'_w = \rho_e(l'_v, x'_e)$. This however means that for each node w there must be at least one incoming edge $e = vw$ in our guess E'_0. This decreases the amount of possible choices for guessing, as we only need to consider all sets $E'_0 \in 2^{E'}$ such that there are no nodes (except s) without incoming edges in E'_0. In the following, whenever we refer to the guessing step, each guess respects this observation, that is we do not guess sets that violate this condition (regardless of guessing right or wrong). We refer to this condition by $(*)$.

Together with the above observation, one could summarize the proposed algorithm in the following pseudocode.

input : $E^* \subseteq E' \subseteq E, u_0 \geq 0$
output: NTF $x' \in K(E', u_0)$ with resetting on E^*, labels l'
for $E'_0 \subseteq E'$ *satisfying* $(*)$ **do**
  if (P) *has an optimal solution* **then**
    $(x', l') \leftarrow$ optimal solution of (P) //Solve
    return x', l'
  end
end

Algorithm 1: Computing an NTF

4.1.1 First interpretation

Not adding aforementioned constraints to the optimization problem could lead to a solution violating the NTF definition, as the following example demonstrates.

Example 4.1. Consider the network from Figure 2.2 given by the graph $G = (V, E')$ where $V = \{s, a, b, t\}$, $E' = \{sa, ab, at, bt\}$ and the set of resetting edges $E^* := \{ab\}$. Clearly, the service rates are properly defined and the pair (E^*, E') satisfies the stated requirements. We set the inflow to a constant

value of $u_0 := 10$.
As explained earlier, an NTF is given by the flow in Figure 2.2a:

$$x'_{sa} = 10, x'_{at} = 6, x'_{ab} = 4, x'_{bt} = 4$$

with labels

$$l'_s = 1, l'_a = l'_b = l'_t = 2.$$

Lemma 2.2 yields that every possible NTF on this network has the exact same labels (in particular: the sum of them is always the same). The sum of these labels equals 7.
Furthermore, an NTF on said network has to send flow along every edge. Sending an amount of 10 flow units either along the path *s-a-t* or *s-a-b-t* will always violate the condition of the definition of an NTF. Thus, the only eligible candidate for E'_0 is the entire set E'. "Guessing" $E'_0 = E'$ then leads to the optimization problem given by:

$$\begin{aligned} \max_{(x',l')} \quad & l'_s + l'_a + l'_b + l'_t \\ s.t. \quad & x' \in K(E'_0, u_0) \\ & l'_s = 1 \\ & l'_a \leq \max\{l'_s, x'_{sa}/5\} \\ & l'_b \leq x'_{ab}/2 \\ & l'_t \leq \max\{l'_a, x'_{at}/7\} \\ & l'_t \leq \max\{l'_b, x'_{bt}/5\}. \end{aligned}$$

Now, consider the flow from Figure 2.2b: $(y'_e)_{e\in E'} \in K(E'_0, u_0)$ defined by sending 10 units along the path *s-a-b-t* and its corresponding labels $g'_s = 1, g'_a = g'_t = 2, g'_b = 5$.
(y', g') is a feasible solution to the above optimization problem. The corresponding objective function value $1 + 2 + 5 + 2 = 10$ is strictly greater than the one of the NTF (x', l'). As every NTF has the same labels, the optimal solution of the above program cannot be an NTF (using the fact that the optimal solutions objective function value has to be at least 10). Clearly, y is not an NTF, since $g'_t = 2 < 5 = \max\{g'_b, y'_{bt}/5\} = \rho_{bt}(g'_b, y'_{bt})$ but $y'_{bt} \neq 0$.

Hence, the constraints in the given optimization problem are not sufficient in order to ensure the correct computation of an NTF. We investigate the second possible interpretation of the algorithm.

4.1.2 Second interpretation

In the following we will thus assume that the authors of [1] intended to add the constraint set $l'_w = \rho_e(l'_v, x'_e)$ for $e = vw \in E'_0$ to the problem formulation. The problem, which has to be solved after guessing E'_0, can be

then formulated as follows.

$$
\begin{aligned}
(P) \qquad \max_{(x',l')} \quad & \sum_{w \in V} l'_w \\
s.t. \quad & x' \in K(E'_0, u_0) \\
& l'_s = 1 \\
& l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e) \text{ for } w \in V \setminus \{s\} \\
& l'_w = \rho_e(l'_v, x'_e) \text{ for } e = vw \in E'_0
\end{aligned}
$$

Before proving some results and explaining the drawbacks of this method, we formulate the problem as a proper mixed integer linear program with a set of binary variables. Said binary variables are needed to circumvent the non-linearity of the maximum function in $\rho_e(l'_v, x'_e)$ for $e = vw \in E' \setminus E^*$. Furthermore, we will use a big M-method approach to realize some of the constraints. $M > 0$ should be chosen big enough to resemble infinity (denoted by $M \simeq \infty$). In our particular setting we can give an explicit formula for M to bound the labels $l'_w, w \in V$. As we have $l'_s = 1$ and $l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$ for every $w \in V \setminus \{s\}$, it is not hard to see that, using an inductive argument, one can choose $M := \max\left\{1, \frac{u_0}{\min_{e \in E} \nu_e}\right\}$ to achieve $0 \leq l'_w \leq M$ for all $w \in V$. To begin with, clearly $l'_s = 1$ is already a suitable constraint for an LP/IP formulation. Furthermore, we enforce the non-negativity of the labels by simply adding the constraint $l'_w \geq 0$ for $w \in V$. Recall that $K(E'_0, u_0)$ is the set of all static s-t-flows $(x'_e)_{e \in E} \geq 0$ of value $u_0 \geq 0$ such that $x'_e = 0$ for $e \notin E'_0$. The constraint $x' \in K(E'_0, u_0)$ can easily be formulated in a linear program using the following constraints.

$$
\begin{aligned}
& x'_e \geq 0 \qquad \forall e \in E'_0 \\
& x'_e = 0 \qquad \forall e \in E' \setminus E'_0 \\
& \sum_{e \in \delta^+(v)} x'_e - \sum_{e \in \delta^-(v)} x'_e = \begin{cases} u_0 & \text{if } v = s \\ 0 & \text{if } v \in V \setminus \{s,t\} \end{cases} \quad \forall v \in V \setminus \{t\}
\end{aligned}
$$

Note that in the well-known LP-Formulations of s-t-flow problems one normally has capacity constraints, which however play no role in the problem of computing an NTF. To improve the readibility of the final optimization problem, we omit these constraints and just write $x' \in K(E'_0, u_0)$.
Let now $e = vw \in E'_0$. We distinguish two cases in order to reformulate the constraint

$$
l'_w = \rho_e(l'_v, x'_e) = \begin{cases} x'_e / \nu_e & \text{if } e \in E^* \\ \max\{l'_v, x'_e / \nu_e\} & \text{if } e \notin E^* \end{cases} .
$$

i) If $e \in E^*$, then $l'_w = \frac{x'_e}{\nu_e}$ can be added to the program.

ii) For each $e \in E'_0 \setminus E^*$ we introduce a binary variable $z_e \in \{0,1\}$ and add the following constraints, which ensure that l'_w attains the maximum

as in the above definition.

$$\begin{aligned} l'_w &\geq l'_v \\ l'_w &\geq x'_e/\nu_e \\ l'_w &\leq l'_v + (1 - z_e)M \\ l'_w &\leq x'_e/\nu_e + z_e M \end{aligned}$$

We can interpret z_e as follows. If $l'_v \geq x'_e/\nu_e$ then $z_e = 1$. Otherwise we have $l'_v < x'_e/\nu_e$, implying that $l'_w = \max\{l'_v, x'_e/\nu_e\} = x'_e/\nu_e$ and thus $z_e = 0$. Last but not least, we model $l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$ for $w \in V \setminus \{s\}$. Clearly, the minimum is already attained for every node w such that there exists an edge $e = vw \in E'_0$, as we require equality to $\rho_e(l'_v, x'_e)$. It is thus sufficient to model the constraint for each edge $e = vw \in E' \setminus E'_0$. Note that $x'_e = 0$ for all of these edges. We distinguish the following two situations.

i) If $e \in E' \setminus (E'_0 \cup E^*)$, then we have $\rho_e(l'_v, x'_e) = \max\{l'_v, 0\} = l'_v$. Thus it is sufficient to add the constraint $l'_w \leq l'_v$.

ii) If $e \in E^* \setminus E'_0$, then $\rho_e(l'_v, x'_e) = x'_e/\nu_e = 0$. Together with the non-negativity we thus have the constraint $0 \leq l'_w \leq 0$, which is equivalent to adding $l'_w = 0$. Note that as $x'_e = 0$, this is the same as the constraint $l'_w = x'_e/\nu_e$.

In total we get the following mixed integer linear program.

$$\begin{aligned} (P) \qquad \max_{(x',l')} \quad & \sum_{w \in V} l'_w \\ s.t. \quad & x' \in K(E'_0, u_0) \\ & l'_s = 1 \\ \\ & l'_w = x'_e/\nu_e \text{ for } e = vw \in E^* \\ \\ & l'_w \geq l'_v \text{ for } e = vw \in E'_0 \setminus E^* \\ & l'_w \geq x'_e/\nu_e \\ & l'_w \leq l'_v + (1 - z_e)M \\ & l'_w \leq x'_e/\nu_e + z_e M \\ \\ & l'_w \leq l'_v \text{ for } e = vw \in E' \setminus (E'_0 \cup E^*) \\ \\ & l'_w \geq 0 \text{ for } w \in V \\ & z_e \in \{0, 1\} \text{ for } e \in E'_0 \setminus E^* \end{aligned}$$

The following results justify the correctness of the algorithm.

Proposition 4.1
Let (x', l') be an NTF of value u_0 with resetting edges $E^* \subseteq E'$ and $E'_0 := \{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\}$.

For $e = vw \in E'_0 \setminus E^*$ *define* $z_e = \begin{cases} 1 & \text{if } \max\left\{l'_v, \frac{x'_e}{\nu_e}\right\} = l'_v \\ 0 & \text{otherwise.} \end{cases}$
Then $(x', l', (z_e)_{e \in E'_0 \setminus E^*})$ *is a feasible solution of (P).*

Proof. We prove the statement by showing that no constraint is violated. Obviously $l'_s = 1$ and $l'_w \geq 0$ for all $w \in V$. Furthermore, $z_e \in \{0, 1\}$ by definition. x' clearly is an s-t-flow of value u_0 and the definition of NTFs yields $x'_e = 0$ for $e \notin E'_0$, proving that $x' \in K(E'_0, u_0)$. Let now $e = vw \in E'$. We distinguish all possible cases.

i) Let $e \in E^*$. Either $e \in E'_0$ and thus $l'_w = \frac{x'_e}{\nu_e}$ or $x'_e = 0$ implying $l'_w = 0$. In any case, the constraint $l'_w = \frac{x'_e}{\nu_e}$ is satisfied.

ii) Let $e \in E'_0 \setminus E^*$. Then, clearly $l'_w = \rho_e(l'_v, x'_e) = \max\left\{l'_v, \frac{x'_e}{\nu_e}\right\} \geq l'_v, \frac{x'_e}{\nu_e}$. Hence, either $l'_w = l'_v$ or $l'_w > l'_v$. In the former case, by definition $z_e = 1$ and clearly both constraints are satisfied, that is $l'_w = l'_v \leq l'_v + 0 = l'_v + (1 - z_e)M$ and $l'_w \leq \frac{x'_e}{\nu_e} + M \simeq \infty$. The latter case works analogously.

iii) Let $e \in E' \setminus (E'_0 \cup E^*)$. Then $l'_w = \min_{\hat{e} = \hat{v}w \in E'} \rho_{\hat{e}}(l'_{\hat{v}}, x'_{\hat{e}}) \leq \rho_e(l'_v, x'_e) = l'_v$.

Thus, all constraints are satisfied, which proves the statement. □

The above proposition implies that if E'_0 has been guessed correctly, then (P) is certainly feasible.

Proposition 4.2
Let $E'_0 \subseteq E'$ *be guessed satisfying condition* $(*)$. *If* (x', l', z) *is a feasible solution of* (P), *then* (x', l') *is an NTF with resetting on* E^*.

Proof. As $x' \in K(E'_0, u_0)$ it remains to show that l' are the labels associated with the flow x' and that (x', l') is an NTF. Clearly, $l'_s = 1$. We begin by showing that for every $w \in V \setminus \{s\}$ the following inequality holds.

$$l'_w \leq \min_{e = vw \in E'} \rho_e(l'_v, x'_e) \tag{15}$$

We distinguish different cases.

i) Let $e \in E^*$. Due to the feasibility, we have that $l'_w = \frac{x'_e}{\nu_e} = \rho_e(l'_v, x'_e)$.

ii) Let $e \in E'_0 \setminus E^*$. Either $z_e = 1$, implying that $l'_w \leq l'_v \leq \rho_e(l'_v, x'_e)$, or $z_e = 0$, which directly implies $l'_w \leq \frac{x'_e}{\nu_e} \leq \rho_e(l'_v, x'_e)$.

iii) Let $e \in E' \setminus (E'_0 \cup E^*)$. By definition and the fact that $l'_v \geq 0$ we have $\rho_e(l'_v, x'_e) = \max\{l'_v, 0\} = l'_v$. The constraints of (P) enforce $l'_w \leq l'_v$, which proves the statement.

It remains to show that (15) holds with equality, that is for every $w \in V \setminus \{s\}$ there exists an edge $e = vw$ such that $l'_w = \rho_e(l'_v, x'_e)$. As E'_0 has been guessed satisfying condition $(*)$ (see Observation 4.1) there exists such an edge.
Finally, we conclude that $l'_w = \min_{e = vw \in E'} \rho_e(l'_v, x'_e)$ for all $w \in V \setminus \{s\}$. (x', l') is an NTF, as for every $e = vw$ with $x'_e > 0$ we have that $e \in E'_0$, as the constraints ensure $x' \in K(E'_0, u_0)$. This however implies that $l'_w = \rho_e(l'_v, x'_e)$ finishing the proof. □

Observe that (P) is always bounded. Formally, for each flow x' with corresponding labels we have $l'_v \leq M$ and thus $\sum_{v \in V} l'_v \leq M|V|$. Hence, (P) can either be infeasible or have an optimal solution. Furthermore, knowing that each NTF has the same labels (see Lemma 2.2) and that by Proposition 4.2 every feasible solution of (P) is an NTF, we have that each feasible solution of (P) must also be optimal. Hence, the objective function of (P) is of no virtue whatsoever and thus the problem of finding an optimal solution of (P) reduces to finding a feasible one. (P) is thus not a proper optimization problem and it would be of no difference to replace the objective function by a constant term.
Even though this interpretation of the proposed method from [1] has this minor deficit, it is still a working approach to compute an NTF in finite time, as the following corollary confirms.

Corollary 4.1
Algorithm 1 terminates and is correct.

Proof. Clearly, the number of iterations is finite. In each iteration, that is for each E'_0, the algorithm has to decide whether an optimal solution (x', l') of (P) exists. As explained above, this problem reduces to deciding whether (P) is feasible. This could, for example, be done in finite time as follows. Given E'_0, guess the set $Z_{E'_0} := \{e \in E'_0 \setminus E^* \mid z_e = 1\}$ and solve $(P_{Z_{E'_0}})$, that is (P) treating z_e as constant. There are exponentially though finitely many possible guesses for $Z_{E'_0}$ and $(P_{Z_{E'_0}})$ is, after replacing the variables z_e with constant values 1 or 0, a linear program, which is solvable in polynomial time using the ellipsoid method (cp. [11]) and in particular in finite time. Observe that there is some $Z_{E'_0}$ such that $(P_{Z_{E'_0}})$ is feasible if and only if (P) is feasible. Thus, the algorithm terminates in finite time.

The correctness is easy to see given Proposition 4.1 as well as Proposition 4.2. If (P) is feasible in some iteration, the algorithm terminates and returns (x', l'), which form an NTF. By Theorem 2.2 there exists an NTF and thus there is at least one E'_0 such that (P) is feasible.

□

4.2 Letting the solver guess

We have seen an approach allowing us to compute an NTF using a routine which tries to find a certain edge set E'_0. As seen before this leads to an exponential number of guesses in the worst case. To speed up the procedure, one could leave the guessing problem to the solver. Modern solvers are capable of many advanced presolving techniques which could give an advantage over "manually" trying all possible subsets. In the following, we present a mixed integer linear programming formulation in which the solver guesses the set

E'_0.

$$
\begin{aligned}
(\hat{P}) \qquad \max_{(x',l')} \quad & \sum_{w \in V} l'_w \\
s.t. \quad & x' \in K(E', u_0) \\
& x'_e \le a_e u_0 \text{ for } e = vw \in E' \qquad (16) \\
& l'_s = 1 \\
\\
& l'_w = x'_e / \nu_e \text{ for } e = vw \in E^* \\
\\
& l'_w \ge l'_v + (a_e - 1)M \text{ for } e = vw \in E' \setminus E^* \\
& l'_w \ge x'_e / \nu_e + (a_e - 1)M \\
& l'_w \le l'_v + (1 - z_e + 1 - a_e)M \\
& l'_w \le x'_e / \nu_e + (z_e + 1 - a_e)M \\
\\
& l'_w \le l'_v + a_e M \text{ for } e = vw \in E' \setminus E^* \\
\\
& l'_w \ge 0 \text{ for } w \in V \\
& z_e \in \{0, 1\} \text{ for } e \in E' \setminus E^* \\
& a_e \in \{0, 1\} \text{ for } e \in E' \\
& \sum_{e \in \delta^-(w)} a_e \ge 1 \text{ for } w \in V \setminus \{s\} \qquad (17)
\end{aligned}
$$

Note that we used the same objective function as before. We could replace it by any other as the following proposition demonstrates. The objective function has been left like this to indicate that this optimization problem has been derived from the last one. Here, the binary variables a_e are intended to model the decision whether $e \in E'_0$ or not. (16) ensures that flow can only be send along edges e that are part of E'_0, that is $a_e = 1$. Note that (17) enforces condition $(*)$.

Proposition 4.3
Every feasible solution of $(\hat{P})$ *yields an NTF and vice versa.*

Proof. It is not hard to see that every NTF (x', l') corresponds to a feasible solution. Let therefore

$$
a_e = \begin{cases} 1 & \text{if } l'_w = \rho_e(l'_v, x'_e) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } e = vw \in E'
$$

and

$$
z_e = \begin{cases} 1 & \text{if } \max\left\{l'_v, \frac{x'_e}{\nu_e}\right\} = l'_v \\ 0 & \text{otherwise} \end{cases} \quad \text{for } e = vw \in E' \setminus E^*.
$$

Clearly, $x' \in K(E', u_0)$ and $l'_s = 1$. Furthermore, as x' is an NTF, the implication $a_e = 0 \Rightarrow l'_w < \rho_e(l'_v, x'_e) \Rightarrow x'_e = 0$ holds. Clearly $x'_e \le u_0$ for all edges. Thus the constraint $x'_e \le a_e u_0$ is satisfied. For $w \in V \setminus \{s\}$ we clearly

have $l'_w \geq 0$ and $l'_w = \rho_e(l'_v, x'_e)$ for some $e = vw$. This implies $a_e = 1$ and thus $\sum_{e \in \delta^-(w)} a_e \geq 1$ is valid.
Let $e \in E^*$. The constraint $l'_w = \frac{x'_e}{\nu_e}$ is obviously valid. Let now $e = vw \in E' \setminus E^*$. We distinguish cases:

i) Let $l'_w = \rho_e(l'_v, x'_e)$. Then by definition $a_e = 1$ and clearly $l'_w \leq l'_v + M \simeq \infty$. Furthermore, the constraints $l'_w \geq l'_v + (1 - a_e)M$ and $l'_w \geq x'_e/\nu_e + (1 - a_e)M$ are valid. Either $l'_w = l'_v$, implying $z_e = 1$ and thus $l'_w = l'_v + 0M$ and $l'_w \leq x'_e/\nu_e + M \simeq \infty$, or $l'_w > l'_v$ which implies $z_e = 0$. It is not hard to see that all constraints hold as well in this case.

ii) Let $l'_w < \rho_e(l'_v, x'_e)$. This is equivalent to $a_e = 0$, implying $x'_e = 0$. Then $l'_w \leq l'_v + 0M$ and also $l'_w \geq l'_v - M, x'_e/\nu_e - M$ as $l'_w \geq 0$. Similarly, either $l'_v = \max\{l'_v, x'_e/\nu_e\}$ or not. In the former case, $z_e = 1$ and then $l'_w \leq l'_v + M \simeq \infty$ as well as $l'_w \leq x'_e/\nu_e + 2M \simeq \infty$. On the contrary, $z_e = 0$ works analogously.

Hence, (x', l', z, a) is a feasible solution.
Let on the other hand (x', l', z, a) be a feasible solution of $(\hat{P})$. $x' \in K(E', u_0)$ and $l'_s = 1$ holds obviously. We show that $l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$. To begin with, if $e \in E^*$ then clearly $l'_w = \rho_e(l'_v, x'_e)$. Let now $e = vw \in E' \setminus E^*$. Again, we distinguish cases.

i) If $a_e = 1$ then similarly to the proof of Proposition 4.2 we have $l'_w = \max\{l'_v, x'_e/\nu_e\}$ which proves the claim.

ii) If however $a_e = 0$ then clearly $l'_w \leq l'_v + 0M = l'_v \leq \rho_e(l'_v, x'_e)$.

Thus it holds that $l'_w \leq \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$. Equation 17 enforces that for each $w \in V \setminus \{s\}$ there is at least one edge $e = vw$ such that $a_e = 1$ and thus $l'_w = \rho_e(l'_v, x'_e)$, implying $l'_w = \min_{e=vw \in E'} \rho_e(l'_v, x'_e)$. Hence, l' are the labels associated to x'. Finally, let $x'_e > 0$. Then $a_e = 1$, implying that $l'_w = \rho_e(l'_v, x'_e)$. (x', l') is thus an NTF. □

This shows that we can compute an NTF by solving one single mixed integer linear program. As discussed later, this works really well on small graphs. The problem with this approach is that the number of binary variables in $\hat{P}$ grows linearly with a factor of 2 in the number of edges E'. In general, solving mixed integer linear problems is NP-hard and thus there is no algorithm known which solves them efficiently. The calculations done in the upcoming sections indicate that the time needed to solve such a programm can dramatically grow with the number of edges in E'.

4.3 Improving the existing approach

In the first part of this section we have given an algorithm that computes an NTF in finite time. Furthermore, we have seen that the objective function of (P) is irrelevant. It thus stands to reason to develop a more efficient approach where the objective function is of actual importance. While it might be desirable to use the objective function in order to relax the binary variables and thus to solve a linear program in each iteration (which is orders

of magnitude faster), there is no obvious way of doing so. It is of interest to not only keep the number of possible E_0' guesses low, but also to have a low number of binary variables in each optimization problem, that needs to be solved. In the following, we try to develop a more efficient approach, which handles this tradeoff.

Examining (P), we observe that E_0' is basically a guess for edges that could potentially send flow, but not necessarily have to. Condition $(*)$ ensures that every node $w \neq s$ has at least one incoming edge from E_0'. This leads to a minimum number of edges in E_0', that is $|E_0'| \geq |V| - 1$ and thus a minimum number of binary variables in each optimization program. The first intuition would be to relax this condition. Consider for example an iteration of algorithm 1 with a guess E_0' that yields a solution, that is, (P) is feasible and returns (x', l'). In many graphs there will be at least one edge $e \in E_0'$ without flow on it. Still $e \in E_0'$ implies that $l'_w = \rho_e(l'_v, x'_e)$. It would be certainly fine to exclude e from E_0' and simply adjust l'_w to still satisfy (15) with equality. It seems likely that this is exactly what the authors of [1] intended with the maximization of $\sum_{w \in V} l'_w$. However, the approach of guessing the set $\{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\}$ (implying condition $(*)$ and consequently the uselessness of the objective function) is to restrictive. We will thus try to guess a set E_0' satisfying $E_0' \subseteq \{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\}$. Still, (P) contains a constraint enforcing that flow is only send along edges from E_0'. Hence, guessing a set E_0' whichs lacks an edge e through which NTFs send flow will lead to an infeasible problem. To summarize, we neglect the approach of guessing $\{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\}$ and instead consider sets E_0' satisfying

$$\{e \mid x'_e > 0\} \subseteq E_0' \subseteq \{e = vw \mid l'_w = \rho_e(l'_v, x'_e)\}. \tag{18}$$

As observed earlier, the number of binary variables in the optimization problems, that need to be solved, depends only on the number of edges in E_0'. Thus, we seek to keep this number as low as possible, in order to solve these programs more efficiently. Consequently, considering (18), it is desirable to try to guess $E_0' := \{e \mid x'_e > 0\}$. Note that the set of edges which have positive flow is not necessarily unique, as there might be different NTFs.

Remark 4.2. Generally, it is not a good idea to somehow enforce the constraint $x'_e > 0$ for $e \in E_0'$ in (P), despite that the solver would probably quickly recognize infeasible instances. First of all, adding $x'_e > 0$ will break the closedness of the feasible region of a linear program. Hence, strict inequality constraints are generally not allowed by LP solvers. One could try to circumvent this problem by adding the constraint $x'_e \geq \varepsilon$ for some small enough constant $\varepsilon > 0$. Apart from the problem of choosing ε correctly, this leads to unwanted behaviour. Even though this approach worked in our computations, NTFs computed by the solver then tend to have one or multiple edges with a flow of exactly ε, leading to numerical instability in further computations regarding the α-extension method.

In order to approach guessing $E_0' = \{e \mid x'_e > 0\}$, a natural condition on E_0' is that for each node $w \in V \setminus \{s, t\}$ the following condition holds.

$$\delta^+_{E_0'}(w) \neq \emptyset \Leftrightarrow \delta^-_{E_0'}(w) \neq \emptyset \tag{19}$$

In other words, we assure that E'_0 is chosen such that each intermediate node w has either both in- and outgoing edges in E'_0 or neither. This ensures that every node is either isolated (within the subgraph induced by E'_0), or it is possible to send flow through said node. Besides, there should be outgoing edges from s and incoming edges to t in E'_0. However, we will shortly see that these two conditions are a side effect of (19). Note that while condition (19) could in general lead to more eligible guesses than condition $(*)$, the benefit over guessing naively (that is, arbitrary subsets of E') is still significant.

To begin with, we make one more assumption. If $u_0 = 0$ then the only NTF is the zero-flow, whose labels can be computed iteratively as E' is acyclic. Therefore we assume that the inflow u_0 is positive. Before computing an NTF we can implement a preprocessing step in order to lower the size of V and E'.

Let $w \in V \setminus \{s, t\}$ such that $\delta^+(w) = \emptyset$, that is no edge in E' goes out from w. Flow will never be send along edges $e = vw$ that arrive in w, simply because this would directly violate the flow conservation constraints. We can safely delete w and all incoming edges of w from the graph. This procedure can be repeatedly executed until there is no such node left. After having computed an NTF on the remaining subgraph, it is fairly easy to construct an NTF on G, as only the labels of deleted nodes have to be adjusted. The following pseudocode describes an iterative approach to delete nodes from G.

while $\exists w \in V \setminus \{s, t\}$ *s.t.* $\delta^+(w) = \emptyset$ **do**
  $V \leftarrow V - w$
  $E' \leftarrow E' \setminus \delta^-(w)$
end
return (V, E')

Algorithm 2: Preprocessing

This approach is motivated as follows. During the iterations of the α-extension method it may happen that an edge leaves the shortest path network. This could in fact lead to entire parts of the network to be cut off from t, that is there might be several nodes that do not lie on a shortest s-t-path anymore. These nodes are irrelevant for the computation of an NTF.

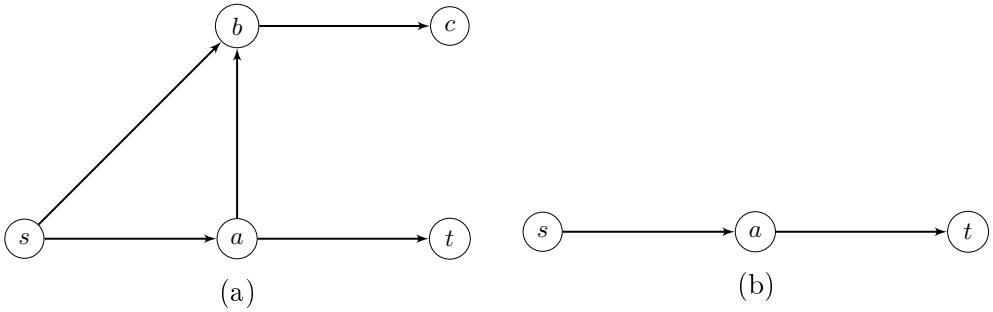


Figure 4.1: A network before and after the preprocessing procedure

Although the idea how to reconstruct an NTF as described above is straightforward, we explain it formally in the next lemma.

Lemma 4.1
Let (V_+, E'_+) be the sets returned from the above preprocessing step. Let further $V_- := V \setminus V_+$. Suppose (x', l') is an NTF of value $u_0 > 0$ with resetting on $E^* \cap E'_+$ in the graph $G_+ = (V_+, E'_+)$. Then an NTF in (V, E') can be found in polynomial time.

Proof. Without loss of generality let $V_- = \{v_1, \ldots, v_k\}$, $k \geq 0$ such that the nodes are sorted in reverse order of their removal. More precisely, algorithm 2 removed the nodes in V_- in the order $v_k, v_{k-1}, \ldots, v_1$. We set $x'_e = 0$ for every $e \in E' \setminus E'_+$
Clearly, v_1 has no incoming edges from nodes in V_-. Consequently for each edge $e = vv_1 \in \delta^-(v_1)$ the label l'_v exists. We can safely set l'_{v_1} to $\min_{e=vv_1} \rho_e(l'_v, x'_e)$. Note that as there is no edge $e = v_1 w \in E'$ with $w \in V_+$, l'_{v_1} value does not violate any other label.
This immediately yields the idea on how to compute all remaining labels. Let now $V_+ := V_+ \cup \{v_1\}, E'_+ := E'_+ \cup \delta^-(v_1)$ and $V_- := \{v_2, \ldots, v_k\}$. Undoubtedly, $v_2, \ldots, v_k$ are still in reverse order of their removal. Repeat the argument until V_- is empty. Eventually we have $l'_w = \min_{e=vw} \rho_e(l'_v, x'_e)$ for all $w \in V, w \neq s$ and $l'_s = 1$. Hence l' are the labels associated with x' and (x', l') is an NTF on $G = (V, E')$, because if $x'_e > 0$ for some $e = vw \in E'$ then e has been in the original E'_+ and thus $l'_w = \rho_e(l'_v, x'_e)$, as the original (x', l') is an NTF in the graph G_+. The fact that this procedure can be completed in polynomial time requires no further explanation. □

In the following assume that V and E' have already been "cleaned up" by the preprocessing algorithm. Whenever we refer to E^* or E', we intend to talk about the edge sets after having removed edges with the above method. The major benefit of applying the preprocessing algorithm is not only that the size of E' is reduced and hence the number of guesses for E'_0, furthermore we also gain more knowledge about how our guesses should be constructed. The following short lemma substantiates this thought.

Lemma 4.2
Let $G = (V, E')$ be such that there is no $w \in V \setminus \{t\}$ with $\delta^+(w) = \emptyset$. Then for all NTFs x' we have that $x'_e > 0$ for all $e \in E^*$.

Proof. Towards a contradiction, suppose there exists a resetting edge $e = vw$ in the remaining graph (remaining in the sense of after the preprocessing step) such that $x'_e = 0$ for some NTF x' with labels l'.
Obviously, then $l'_w = 0$. But as x' is an NTF, this implies that there cannot be an edge $\hat{e} = \hat{v}w$ in E' such that $x'_{\hat{e}} > 0$. This is simply because any such edge would violate the definition of NTFs, as $l'_w = 0 < \rho_{\hat{e}}(l'_{\hat{v}}, x'_{\hat{e}})$. By our assumption there exists a path $P = w\text{-}v_1\text{-}v_2\text{-}\ldots\text{-}v_k\text{-}t$ from w to t (the graph is acyclic and each node except t must have an outgoing edge). By the flow conservation constraints and the fact that there is no flow going into w, clearly there is no flow going out of w. Thus $l'_{v_1} = 0$. Repeating the above argument along P leads to the conclusion that $l'_t = 0$ and that there is no flow arriving in t, contradicting the assumption that x' is an s-t-flow of value $u_0 > 0$. □

Formulated differently, it would make no sense to guess a set E'_0 that does not contain some edge $e \in E^*$, as this would directly imply that no flow can be send along that edge. Thus, the following inclusion should hold for all guesses:

$$E^* \subseteq E'_0. \tag{20}$$

Obviously, this reduces the number of possible guesses even further, assuming that $E^* \neq \emptyset$. More precisely, the number of guesses in the naive approach, which is $2^{|E'|}$, is reduced by a factor of $2^{|E^*|}$. In the case that there are no resetting edges, we can simply compute an NTF in polynomial time using the approach from [8]. Hence, we assume that $E^* \neq \emptyset$. Note that given (20) and this assumption, (19) ensures that $\delta^+_{E'_0}(s) \neq \emptyset \neq \delta^-_{E'_0}(t)$. This is due to the acyclicity of (V, E').

We can now state improved versions of Proposition 4.1 and Proposition 4.2. Note that this time, the objective function $\max \sum_{w \in V} l'_w$ is indispensable.

Proposition 4.4
Let (x', l') be an NTF with resetting edges $E^* \subseteq E'$ and $E'_0 := \{e \mid x'_e > 0\}$. For $e = vw \in E'_0 \setminus E^*$ define $z_e = \begin{cases} 1 & \text{if } \max\left\{l'_v, \frac{x'_e}{\nu_e}\right\} = l'_v \\ 0 & \text{otherwise.} \end{cases}$
Then $(x', l', (z_e)_{e \in E'_0 \setminus E^*})$ is a feasible solution of (P).

We omit the proof, as it is almost identical to the one of Proposition 4.1.

Proposition 4.5
Let $E'_0 \subseteq E'$ be guessed satisfying condition (19). If (x', l', z) is an optimal solution of (P), then (x', l') is an NTF with resetting on E^*.

Proof. The first part of this proof is similar to the one of Proposition 4.2. We have that $x' \in K(E'_0, u_0)$ and $l'_s = 1$. Moreover, it can analogously be proved that (15) holds. Hence, it remains to show that (15) is satisfied with equality. Fix $w \neq s$. Without loss of generality we assume that w is isolated in the subgraph induced by the edges E'_0. If this is not the case, then (19) implies that there must exist some edge $e = vw \in E'_0$, implying $l'_w = \rho_e(l'_v, x'_e)$, which proves the statement. Hence, using that $E^* \subseteq E'_0$, for all $e = vw \in E'$ we have $e \in E' \setminus (E'_0 \cup E^*)$ and thus $l'_w \leq l'_v$. Towards a contradiction, suppose now that (15) is not satisfied with equality for w. Stated differently, we have $l'_w < l'_v$ for all $e = vw$. The idea now is as follows. We increase l'_w to $\min_{e=vw \in E'} l'_v$ and show that the new label does not violate any constraints, implying that we have found another feasible solution with higher objective function value, contradicting the optimality of (x', l', z). Clearly, the inequality $l'_w \leq l'_v$ is still valid for $e = vw$. It is sufficient to consider all outgoing edges from w. Let thus $\hat{e} = w\hat{v}$. We assumed w to be isolated in E'_0, hence we have $\hat{e} \in E' \setminus (E'_0 \cup E^*)$. The only constraint that needs to be satisfied is thus that $l'_{\hat{v}} \leq \min_{e=vw \in E'} l'_v$. But this is clearly the case as $l'_{\hat{v}} \leq l'_w < \min_{e=vw \in E'} l'_v$. This contradicts the assumption.
Finally, let $e = vw \in E'$ with $x'_e > 0$. Obviously this implies $e \in E'_0$. The constraints enforce $l'_w = \rho_e(l'_v, x'_e)$, proving that (x', l') is an NTF. □

Hence, algorithm 1 is correct and terminates if we replace condition $(*)$ by condition (19).

5 Implementation

Part of this thesis has been the implementation of the aforementioned methods to compute normalized thin flows with resetting and consequently Nash flows over time, embedded in a graphical user-interface framework allowing not only quick and easy creation of networks but also the analysis of e.g. earliest arrival time functions, NTFs and inflow functions in a dynamic equilibrium, which can be computed with a single click for further research. The project, called *NashFlowComputation* (short: NFC), has been written in Python 2.7.12 and consists of over 4000 lines of code. It has been designed to work on Linux-based operating systems but can be easily extended to work on other operating systems as Windows and macOS, given that they allow the installation of the third-party software and modules needed for NFC to run. NetworkX 1.11 provides the necessary data structures and algorithms for directed graphs. The computation of Nash flows over time heavily relies on the solving of mixed 0-1 linear programs. NFC currently supports only one solver, namely SCIP 4.0.0, together with the non-commercial LP-solver SoPlex 3.0.0. Furthermore, we used the modeling language ZIMPL to formulate templates for optimization problems. Last but not least, the entire GUI has been coded in PyQT4 and designed using QT-Designer.
It is worth noting that the actual implementation of algorithms discussed in this thesis accounts for a small part of the entire software. Until this moment there has been no popular Python library allowing a UI-based graph or network creation. Therefore, a lot of effort has been put in the development of an adequate class structure, allowing the creation of directed graphs, while using the features of Matplotlib to display the graphs in a visually appealing manner. The graph creation canvas of NFC makes it possible to easily add and remove nodes and edges as well as to assign capacity or transit-time values. Furthermore, zooming in and out as well as moving the display area are included features. Although NetworkX provides methods to plot directed graphs using Matplotlib, these are not sufficient in order to display edges similar to Figure 2.1. These methods have been re-designed from scratch. The software can be cloned from the git-repository https://gitlab.tu-berlin.de/m.zimmer/NashFlowComputation.

5.1 Requirements

We list required third party software and modules and give brief explanations if necessary as well as links to said software.

- Python 2.7.12 - https://www.python.org
- SCIP Optimization Suite - http://scip.zib.de - Includes:
  - SCIP 4.0.0
  - SoPlex 3.0.0
  - ZIMPL 3.3.4
- PyQT 4.12.1 - https://riverbankcomputing.com

- NetworkX 1.11 - https://networkx.github.io
- Numpy 1.11 - http://www.numpy.org - Python scientific computing library including data structures and methods for arrays, matrices and vectors.
- Matplotlib 1.5.1 - http://matplotlib.org - Python plotting library. Used to display the networks and to handle most of the user input as well as the animation of Nash flows over time.

5.2 File structure

To properly use NFC and to be able to further develop or enhance existing methods, one has to understand how the code is structured. Firstly, the root directory contains only one script file which does not need a lot of explanation:

- mainControl.py - This is the script that the user has to start. It basically manages the initialization of the graphical user interface.

The root directory contains a subfolder named source, including most of the important classes and scripts, which are listed below.

- application.py - Responsible for coordination and management of the user interface. Contains the class Interface, which handles signals and communication between different widgets, i.e. components of the GUI. If e.g. a button is pressed, application.py handles the signal and calls the appropriate functions.
- plotCanvasClass.py - Contains the class PlotCanvas, which lays the foundation for the creation of graphs. Handles mouse clicks and releases, zooming and provides the functionality to drag and drop. This class is the base for other classes.
- plotAnimationCanvasClass.py - This file contains the class PlotAnimationCanvas, which inherits and extends the class PlotCanvas in order to provide the tools to compute and visualize the flow animation.
- plotNTFCanvasClass.py - Similarly to plotAnimationCanvasClass.py, this file contains a class extending PlotCanvas in order to properly display normalized thin flows with resetting.
- plotValuesCanvasClass.py - Provides functionality to easily plot real valued functions, e.g. earliest arrival time functions of nodes or cumulative inflow/outflow of edges.
- normalizedThinFlowClass.py - The class NormalizedThinFlow manages all methods and data needed for a thin flow. The rule of thumb here is that each instance corresponds to one solved mixed integer linear program.
- flowIntervalClass.py - Class named FlowInterval managing one extension as described in section 3.

- nashFlowClass.py - The class NashFlow maintains a list of several FlowInterval instances and coordinates the computation of the dynamic equilibria.
- utilitiesClass.py - Contains a variety of static methods used in other classes.

The *source* directory contains two subdirectories, namely *templates* and *ui*. The *ui* directory contains two files.

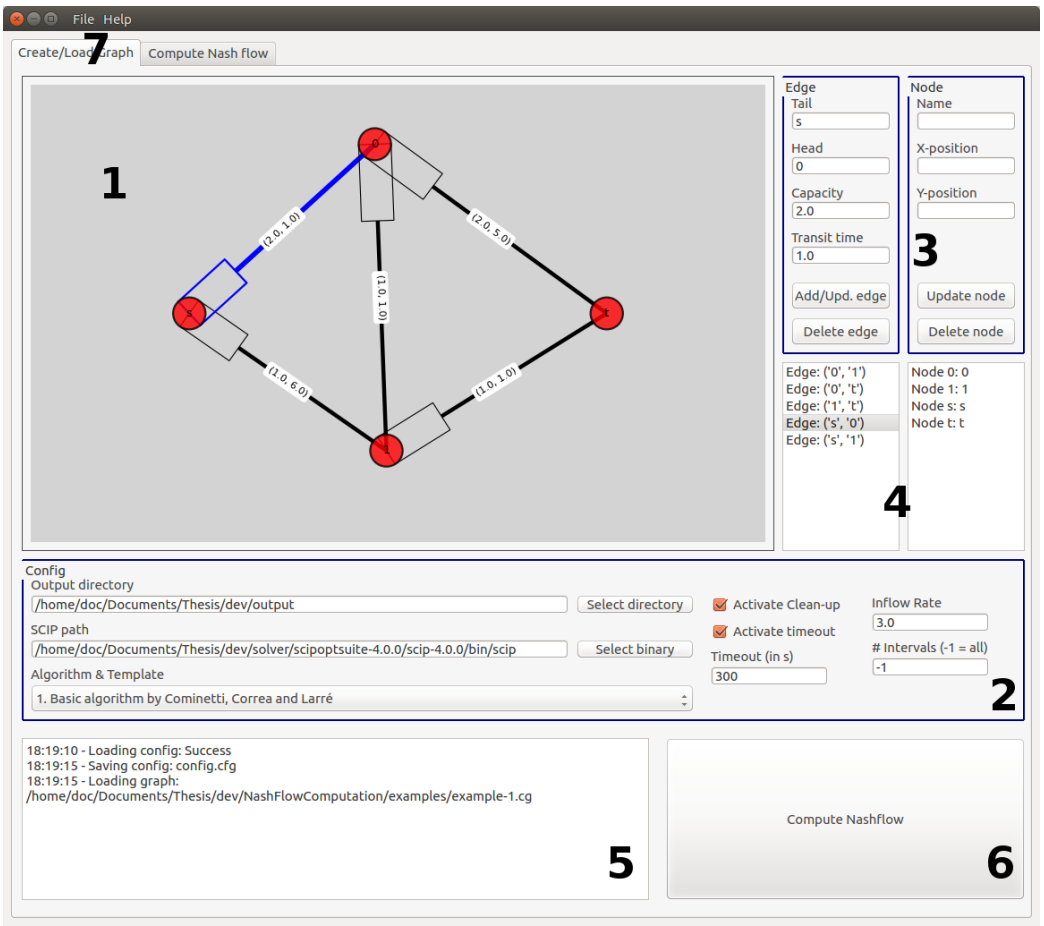
- mainWdw.ui - QT-Designer Project File
- mainWdw.py - Automatically generated file from mainWdw.ui.

The folder *templates* contains three ZIMPL-files (.zpl), one for each method from section 4. These templates are used to model the mixed integer linear programs.

5.3 Documentation

The graphical user-interface is subdivided into two tabs. One tab for the creation of networks and a second one for the visualization of computed Nash flows over time. Below are screenshots of these tabs with explanations of their basic functionality.

Tab 1: Graph creation and configuration



1. The canvas allowing a visualized creation of graphs. Nodes are drawn as red points, edges as black lines with boxes at the beginning, which resemble the queues (see Figure 2.1). The currently selected node or edge is highlighted in blue. Elements of the canvas are selectable with a left click. Selected elements can easily be removed by pressing the Delete key. Labels on edges indicate the capacity ν_e and transit time τ_e in that order. While nodes can be created by left-clicking into the empty (grey) space, edges are created via dragging and dropping between nodes. Pressing the right mouse button above a node allows to drag the node around. Furthermore, scrolling simply zooms in or out and holding mouse 3 allows to move around the field of view.

2. This is the groupbox containing all important details regarding the configuration. The user has to specify an output directory (where the solvers output logs are saved), as well as the path to the SCIP binary. Furthermore, the user can select between all three previously introduced methods in order to compute NTFs. If the "Activate Clean-up"-checkbox is selected, then all SCIP logs are automatically deleted from the output directory. If after a certain amount of time the NTF computation should be aborted, a timeout can be specified. Finally, the number of α-extension intervals that should be computed can be specified (inserting -1 is equivalent to computing all intervals, that is until the upper bound of some interval is ∞).

3. Manual modification of edge and node properties. Can be also used to create edges.

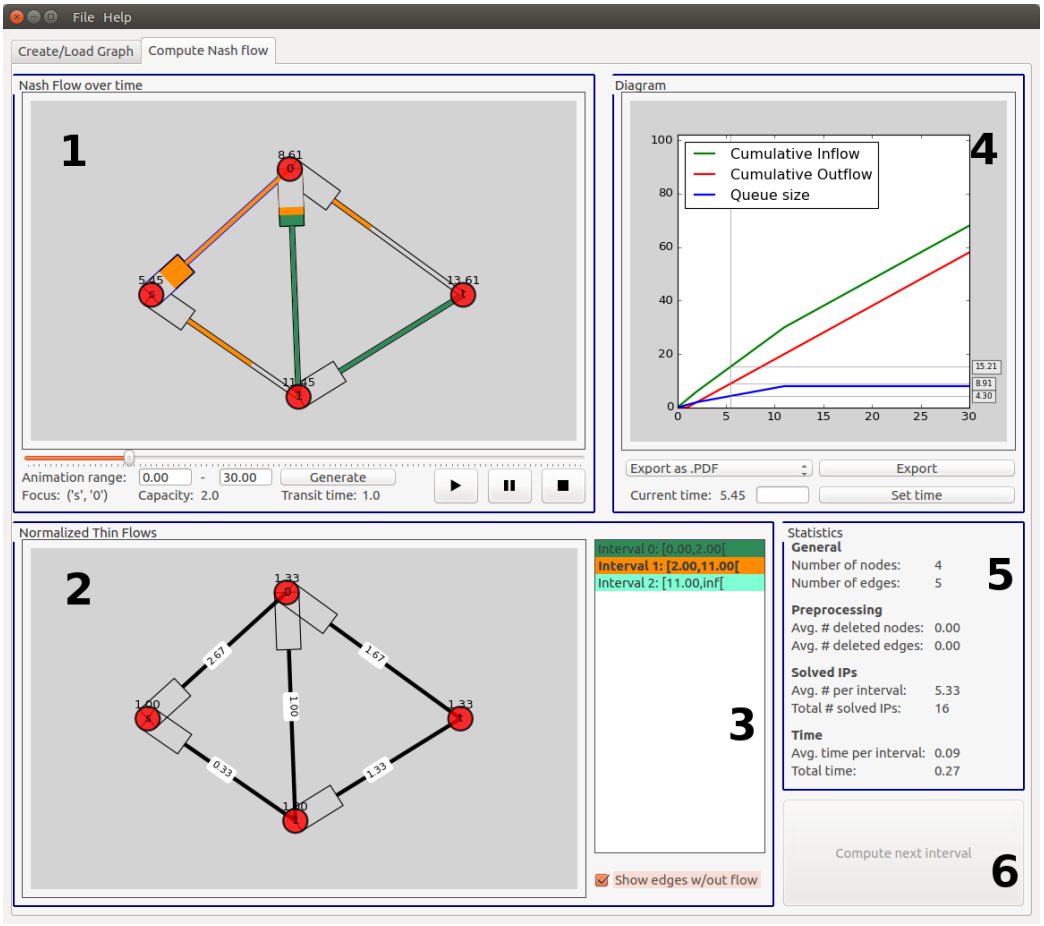
4. A list of all nodes and edges. Clicking on an entry is equivalent to selecting the respective element in the canvas.

5. A log containing all executed actions.

6. Button to start the computation of a Nash flow over time.

7. Via the "File"-menu, the user can clear the graph creation canvas as well as load and save graphs.

Tab 2: Nash flow animation and analysis



1. Canvas to display Nash flows over time. The slider below can be moved manually or automatically by pressing the PLAY-button. When the slider moves, the canvas displays a different frame of the animation over time. NTFs are colored differently and flow through the network. Similar to the canvas in the first tab, edges and nodes are selectable. A selected element has a plot of its values displayed in canvas 4.

2. Displays the NTF to which the animations currently selected point in time belongs. Edges without flow in a specific NTF can be visualized or hidden.

3. List of all so far computed NTFs and α-extension intervals. Clicking on one of them, changes the NTF display in 3.

4. A canvas displaying plots of certain functions. If a node v has been selected, then the canvas displays the node label function l_v. On the other hand, if an edge e has been selected, the canvas displays the corresponding queue size z_e as well as the cumulative in- and outflow F_e^+, F_e^-. The plot is clickable. If the user clicks inside it, the interface jumps to the corresponding time point.

5. Statistics of the last Nash flow computation.

6. If not all intervals have been computed already, then this button is enabled, allowing to compute the next interval.

5.4 Computational study

While the discussed improvements in section 4.3 are merely of heuristical nature and do not guarantee a more efficient computation of Nash flows over time, it is still of interest to compare the methods with respect to the time needed to solve sample instances. We experimentally compare the running time of the three previously discussed methods to compute normalized thin flows with resetting. In the following we denote the methods as follows.

- Method 1(M1): The approach from [1] explained in section 4.1.2.
- Method 2(M2): Method 1 but the solver determines E_0' (see section 4.2).
- Method 3(M3): The improved method from section 4.3.

While in M2 only one mixed integer linear program is solved, M1 and M3 are both implemented with different backtracking approaches in order to determine their respective edge set E_0'. To be able to compare the methods in a fair setting, we carried out all computational experiments on the same hardware, a Lenovo Thinkpad X230 with Intel Core i5-3320M (2.60 GHz) processor and 8GB RAM. A list of the networks we created and on which we computed the entire Nash flow over time with a timeout of 9000 seconds can be found in Table 1. We describe Table 2, which displays the computational results.

Network	# Nodes	# Edges	# Intervals
1	4	5	3
2	9	15	6
3	32	57	4
4	22	40	3
5	33	57	1
6	22	40	1
7	22	42	1

Table 1: The 4th column indicates the number of α-extension intervals computed. Network 1 is the network from [8] which has also used been in subsection 5.3. Note that all NTF implementations work on the shortest path networks given certain times. These are usually smaller than the original networks.

Table 2: Computational results

	Time	∅ IPs	∅ Binary vars./IP	∅ Del. nodes	∅ Del. edges
1;M1	0.31	5.33	3.50	0	0
1;M2	0.12	1	7.33	0	0
1;M3	0.17	2.33	3.86	0	0
2;M1	5.14	54	9.46	0	0
2;M2	0.10	1	18.5	0	0
2;M3	0.20	2.00	10.33	0.33	0.67
3;M1	4.32	56	32.53	0	0
3;M2	0.31	1	70.75	0	0
3;M3	0.13	1.25	4.20	26.25	30.25
4;M1	0.45	8.33	21.92	0	0
4;M2	-	-	-	-	-
4;M3	1.29	12.67	22.63	0	0
5;M1	-	-	-	-	-
5;M2	-	-	-	-	-
5;M3	0.23	14.00	20.57	14.00	22.00
6;M1	0.48	12.00	21.00	0	0
6;M2	0.54	1	80.00	0	0
6;M3	0.08	3.00	8.00	0	0
7;M1	-	-	-	-	-
7;M2	0.59	1	84.00	0	0
7;M3	0.48	22.00	24.68	0	0

The entries of the first column are of the form i;Mj where i indicates the network and $j \in \{1, 2, 3\}$ indicates the method used. While the second column shows the total time in seconds needed to compute all α-extension intervals, the remaining columns display averaged values. Columns 3, 5 and 6 show the average number of solved integer programs, deleted nodes and deleted edges within the preprocessing step (which is only implemented in M3 obviously, thus all other rows evaluate to 0). While we average in those three columns over all α-extension intervals, column 4 shows the average number of binary variables per solved IP. Unsurprisingly, the average number of solved IPs of M2 is always equal to 1. Rows containing only hyphens correspond to instances that timed out, i.e. the solver was not able to compute the NTFs in less than 9000 seconds.

We give interpretations of the experimental results. Clearly, M2 dominates the other methods on small networks. A reason might be that the entire problem is given to the solver. SCIP contains many advanced presolving techniques which simplify the problem. Furthermore, SCIP is implemented in C which tends to be considerably faster than Python. This might be also of importance because the entire backtracking procedure from M1 and M3 is avoided. Interestingly, the performance of method 2 seems to be more dependent upon the underlying structure instead of the size of the instances.

Consider e.g. the equally-sized instances 4 and 6. While M2 was not able to compute a Nash flow over time in instance 4, the other methods performed quite good. However, instance 6;M2 was computed in a reasonable amount of time. As expected, method 3 often leads to both a small number of binary variables per IP and a small number of IPs that need to be solved. In all instances except instance 4, M3 outperformed M1. Network 3 as well as 5 show the efficiency of the preprocessing step. While in network 5 the methods 1 and 2 fail, M3, which reduced the size of the shortest path network by 14 nodes and 22 edges on average, solves the instance in roughly 0.20 seconds. In network 7, the importance of the previously discussed tradeoff between binary variables and number of IPs becomes visible. M3 outperforms M2 by solving 22 IPs with an average of 24.68 binary variables per IP, instead of one single IP with 84 binary variables. Still, the results of network 4 emphasize that the ideas from M3 are heuristical. Which method performs best is highly dependent on the network instance and it is certainly possible to construct graphs that lead to different results.

6 Conclusion and outlook

Although we provided a method that seems to often compute NTFs faster than the existing approach from [1], this approach is still far from being efficient. In general, the number of possible edge sets E'_0 is still exponential and furthermore solving mixed integer linear problems is NP-hard. While Koch and Skutella showed that the efficient computation of NTFs without resetting is possible, the question whether NTFs with resetting are computable in polynomial time remains open and the complexity of the problem is an interesting topic for further research. Cominetti, Correa and Larré conjectured that the problem of finding an NTF might be solvable in polynomial time. Additionally, it is of particular interest for the computational complexity of Nash flows over time whether it is possible to bound the number of α-extension intervals. At this time there are no results regarding the question whether there exist instances such that there are infinitely many such intervals.

Another promising direction of research could be the development of methods to compute Nash flows over time for more complex inflow functions u. The method from [8] works only if u is (piecewise) constant. A generalization to e.g. (piecewise) linear inflow rate functions would certainly be of interest. The field of research of Nash flows over time undoubtedly poses many interesting questions. With the software from section 5 we provided useful and extensive tools to make further research easier.

A Technical details

In the following, we define some of the functional spaces needed for a precise definition of flows over time. Apart from Lemma A.4, we only give proofs for results we did not find in secondary literature. Most of the other results are taken from [10].

Within this thesis, we often use the term *almost everywhere*. A certain property is said to be satisfied *almost everywhere* (short: a.e.) if the set for which the property does not hold, is a set of measure zero.

We denote by $AC(I)$ the space of all *absolutely continuous* functions $u : I \to \mathbb{R}$ and introduce the concept of *local absolute continuity* as follows: u is called *locally absolutely continuous*, if u is absolutely continuous in $[a, b]$ for every interval $[a, b] \subset I$. Similarly, we will by $AC_{loc}(I)$ refer to the space of all locally absolutely continuous functions u. We omit a formal definition of absolute continuity (which can be found in [10]), as the following theorems and propositions sufficiently characterize it.
First however, we introduce two more spaces. The first one is the well-known vector space $L^1(I)$ of Lebesgue-measurable functions $u : I \to \mathbb{R}$ such that $|u(\cdot)|$ is integrable on I, followed by $L^1_{loc}(I)$, similarly containing all measurable functions u satisfying the property of $|u(\cdot)|$ being integrable on every bounded interval (which we call *locally integrable*).

In chapter 3 of Leoni (2009) [10] an important characterization of local absolute continuity has been proven.

Theorem A.1 (Fundamental theorem of calculus)
Let $I \subset \mathbb{R}$ be an interval. A function $u : I \to \mathbb{R}$ belongs to $AC_{loc}(I)$ if and only if

i) u is continuous in I

ii) u is differentiable almost everywhere in I and $u' \in L^1_{loc}(I)$

iii) the fundamental theorem of calculus is valid, i.e. for all $x, x_0 \in I$: $u(x) = u(x_0) + \int_{x_0}^{x} u'(t)dt$.

A better understanding of $AC_{loc}(I)$ can be reached with the following two lemmata.

Lemma A.1
Let $I \subset \mathbb{R}$ be an interval and let $u : I \to \mathbb{R}$ be a Lebesgue integrable function. Fix $x_0 \in I$ and let $U(x) := \int_{x_0}^{x} u(t)dt$ for $x \in I$. Then $U \in AC(I)$ and $U'(x) = u(x)$ for almost all $x \in I$.

Lemma A.2
Let $u, v \in AC_{loc}(\mathbb{R})$. Then the following holds.

i) $u \pm v \in AC_{loc}(\mathbb{R})$ and

ii) $u \cdot v \in AC_{loc}(\mathbb{R})$ and

iii) Let u be monotone. Then $v \circ u \in AC_{loc}(\mathbb{R})$

The following is a corollary of Theorem A.1, which is needed to establish the FIFO-property of the edge-queue model.

Corollary A.1
Let $u : I \to \mathbb{R}$ belong to $AC_{loc}(I)$ such that $u'(x) \geq 0$ for almost all $x \in I$. Then u is nondecreasing.

Proof. Let $x_0, x, x' \in I$ such that $x \leq x'$. Theorem A.1 yields the claim, as the following inequality holds:

$$u(x') - u(x) = \int_{x_0}^{x'} u'(t)dt - \int_{x_0}^{x} u'(t)dt = \int_{x}^{x'} \underbrace{u'(t)}_{\geq 0 \text{ a.e.}} dt \geq 0.$$

□

To derive more useful properties of $AC_{loc}(\mathbb{R})$ functions, we recall the definition of local Lipschitz-continuity, referring to chapter 7 and 13 of Königsberger [9].

Theorem A.2 (Superposition)
Let $f : \mathbb{R} \to \mathbb{R}$. Then the following equivalence holds.

$$f \circ u \in AC_{loc}(\mathbb{R}) \text{ for all } u \in AC_{loc}(\mathbb{R}) \Leftrightarrow f \text{ is locally Lipschitz}$$

Lemma 2.1 requires the following corollary.

Corollary A.2
Let $f, g \in AC_{loc}(\mathbb{R})$. Then $\min\{f, g\} \in AC_{loc}(\mathbb{R})$.

Proof. Clearly, for $x \in \mathbb{R}$ we have the following equality.

$$\min\{f, g\}(x) = \frac{1}{2}(f(x) + g(x)) - \frac{1}{2}|f(x) - g(x)|$$

We prove that, given a function $h \in AC_{loc}(\mathbb{R})$, the composition $|h|$ is also locally absolutely continuous. As an immediate consequence of the triangle-inequality, $|\cdot|$ is Lipschitz-continuous and thus locally Lipschitz. Hence Theorem A.2 yields the desired result. Furthermore, using Lemma A.2, $f - g \in AC_{loc}(\mathbb{R})$ and thus also $|f - g|$. We can conclude that the claim of the corollary is true, as $\min\{f, g\}$ is simply a composition of $AC_{loc}(\mathbb{R})$ functions. □

The following lemma gives us a measure-theoretical insight, which is needed in the proof of the characterization of dynamic equilibria. A proof can be found in [1].

Lemma A.3
Let $g : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a locally integrable function, i.e., $g \in L^1_{loc}(\mathbb{R})$ and let $\{]a_i, b_i[\}_{i \in I}$ a possibly uncountable family of open (or semi-open) intervals. Then g vanishes almost everywhere on each $]a_i, b_i[$ if and only if g vanishes almost everywhere on $\bigcup_{i \in I}]a_i, b_i[$.

The next corollary is a simplification of Theorem 3.54 from [10]. The original theorem is much stronger, but this corollary is sufficient.

Corollary A.3 (Change of Variables)
Let $h \in AC_{loc}(\mathbb{R})$ be monotone and $g : \mathbb{R} \to \mathbb{R}_{\geq 0}$ locally integrable, then the change of variable formula holds:

$$\int_{h(a)}^{h(b)} g(\xi)d\xi = \int_a^b g(h(y))h'(y)dy.$$

Corollary A.4 (Chain Rule)
Let $f, h \in AC_{loc}(\mathbb{R})$ and h monotone. Then for almost all θ, the chain rule holds:

$$(f \circ h)'(\theta) = f'(h(\theta))h'(\theta).$$

To prove the existence of NTFs, the subsequent definition and fixed point theorem are needed(see [6]).

Definition A.1. Let $\Gamma : K \to 2^K$ be a set-valued map. Given two converging sequences $(x^n), (y^n)$ in K with limit x^* and y^* such that $y^n \in \Gamma(x^n)$, we say that the graph $\{x, y \mid y \in \Gamma(x)\}$ is closed if $y^* \in \Gamma(x^*)$.

Theorem A.3 (Kakutanis Fixed Point Theorem)
Let K be a closed, bounded, convex set in $\mathbb{R}^N$ and $\Gamma : K \to 2^K$. For every $x \in K$ let $\Gamma(x)$ be non-empty and convex. Assume the graph $\{x, y \mid y \in \Gamma(x)\}$ is closed. Then some point x^* lies in $\Gamma(x^*)$.

Last but not least, we give the missing proof of Lemma 2.1 as well as an auxiliary lemma, used in Theorem 2.1.

Proof of Lemma 2.1. We prove each claim separately.

i) Recall that $T_{e_1}, \ldots, T_{e_j}$ are locally absolutely continuous and nondecreasing. Hence, l^P is also nondecreasing. Furthermore, using the monotonicity of $T_{e_k} \circ \ldots \circ T_{e_1}, 1 \leq k \leq j$ and Lemma A.2 (iii) it becomes obvious that

$$l^P = T_{e_j} \circ \ldots \circ T_{e_1} \in AC_{loc}(\mathbb{R}).$$

Let now $P_1, ..., P_i$ be the (finitely many) s-w-paths in G. Observe that we can rewrite l_w as $l_w(\theta) = \min\left\{l^{P_1}(\theta), \ldots, l^{P_i}(\theta)\right\}$. The claim thus follows from repeated application of Corollary A.2.

ii) Suppose that there is a cycle $C = (e_1, \ldots, e_j)$ in G_θ with $e_i = v_i v_{i+1}$ for $i \in \{1, \ldots, j-1\}$ and $e_j = v_j v_1$. As all edges of C are in the θ-shortest-path graph, the following equation holds.

$$l_{v_1}(\theta) = T_{e_j}(l_{v_j}(\theta)) = T_{e_j}(T_{e_{j-1}}(l_{v_{j-1}}(\theta))) = \ldots = T_{e_j} \circ \ldots \circ T_{e_1}(l_{v_1}(\theta))$$

But as $q_{e_i}(\theta') \geq 0$ for all $i \in \{1, \ldots, j\}$ and times $\theta' \in \mathbb{R}$, the following inequality is valid too, contradicting the existence of C by using our

previous assumption that every cycle in G has positive free flow transit time.

$$l_{v_1}(\theta) = T_{e_j} \circ \ldots \circ T_{e_1}(l_{v_1}(\theta)) \geq l_{v_1}(\theta) + \underbrace{\sum_{e \in C} \tau_e}_{>0} > l_{v_1}(\theta)$$

Thus G_θ is acyclic. As $l_w(\theta)$ is the earliest arrival time at w we clearly have that $l_s(\theta) = \theta$. Furthermore, as observed earlier, we have that $T_e(l_v(\theta)) \geq l_w(\theta)$ for $e = vw \in E$. For every node $w \in V \setminus \{s\}$ there exists at least one edge $e = vw \in E$ such that $T_e(l_v(\theta)) = l_w(\theta)$, simply because there exists a shortest s-w-path in G_θ. This concludes the proof.

□

The following is due to [1].

Lemma A.4
Let $e = vw$ and $\theta \in \mathbb{R}$. Let $\theta' \leq \theta$ be the largest time such that $T_e(l_v(\theta')) = l_w(\theta)$ and define $I_\theta :=]\theta', \theta]$. Then we have the following set equality.

$$\Theta_e^c = \bigcup_\theta I_\theta$$

Proof. First of all, we have to justify that θ' is properly defined. In general, it holds that $T_e(l_v(\theta)) \geq l_w(\theta)$. Furthermore, we have that $\lim_{\theta' \to -\infty} T_e(l_v(\theta')) = -\infty$. This is justified by the following argument. Observe that for $\theta' \leq 0$, all queues are empty. Hence the travel time through an edge e can be expressed as $T_e(\theta') = \theta' + \tau_e$. Given a path $P = (e_1, \ldots, e_j)$ this however implies that the arrival time at the end of P, that is $l^P(\theta')$, is simply given by the sum of θ' and the constant sum of the free transit times on P, that is $\sum_{e \in P} \tau_e$. Consequently, we have that

$$\lim_{\theta' \to -\infty} l^P(\theta') = \lim_{\theta' \to -\infty} (\theta' + \sum_{e \in P} \tau_e) = -\infty,$$

which implies

$$\lim_{\theta' \to -\infty} l_v(\theta') = -\infty$$

and the claim, using that $q_e(l_v(\theta')) = 0$ for $\theta' \leq 0$ and thus $T_e(l_v(\theta')) = l_v(\theta') + \tau_e$. Hence, the desired θ' exists. We prove the statement by showing that the sets include each other.
"$\subseteq$" Let $\theta \in \Theta_e^c$, that is $T_e(l_v(\theta)) > l_w(\theta)$.
Thus, $\theta' < \theta$ and $\theta \in I_\theta \subseteq \bigcup_{\theta''} I_{\theta''}$.
"$\supseteq$" Let $\theta'' \in I_\theta$ for some θ. As $\theta'' > \theta'$, the definition of θ' and the monotonicity of T_e directly imply that

$$T_e(l_v(\theta'')) > T_e(l_v(\theta')) = l_w(\theta) \geq l_w(\theta'').$$

We conclude $\theta'' \in \Theta_e^c$, which proves the claim. □

References

[1] Roberto Cominetti, José Correa, and Omar Larré. Dynamic equilibria in fluid queueing networks. *Oper. Res.*, 63(1):21–34, February 2015.

[2] Roberto Cominetti, José Correa, and Neil Olver. *Long Term Behavior of Dynamic Equilibria in Fluid Queuing Networks*, pages 161–172. Springer International Publishing, Cham, 2017.

[3] Roberto Cominetti, José R. Correa, and Omar Larré. *Existence and Uniqueness of Equilibria for Flows over Time*, pages 552–563. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[4] L. Fleischer and Éva Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1996.

[5] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Rand Corporation research study. University Press, 1962.

[6] J. Franklin. *Methods of Mathematical Economics*. Society for Industrial and Applied Mathematics, 2002.

[7] David Gale. Transient flows in networks. *Michigan Math. J.*, 6(1):59–63, 1959.

[8] Ronald Koch and Martin Skutella. Nash equilibria and the price of anarchy for flows over time. *Theory of Computing Systems*, 49(1):71–97, 2011.

[9] K. Königsberger. *Analysis 1*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2003.

[10] G. Leoni. *A First Course in Sobolev Spaces*. Graduate studies in mathematics. American Mathematical Soc., 2009.

[11] Martin Skutella Martin Henk. Skript Geometrische Grundlagen der Linearen Optimierung, TU Berlin, 2015.

[12] Martin Skutella. *An Introduction to Network Flows over Time*, pages 451–482. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.