WORKING PAPER

ITLS-WP-12-03

Quasi-dynamic network loading: Adding queuing and spillback to static traffic assignment.

By

Michiel Bliemer, Luuk Brederode[1] , Luc Wismans[1,3] & Erik-Sander Smits[2]

[1] Goudappel Coffeng BV, The Netherlands
[2] Delft University of Technology, The Netherlands
[3] University of Twente, The Netherlands

February 2012

ISSN 1832-570X

INSTITUTE of TRANSPORT and LOGISTICS STUDIES

The Australian Key Centre in Transport and Logistics Management

The University of Sydney

Established under the Australian Research Council's Key Centre Program.

NUMBER: Working Paper ITLS-WP-12-03

TITLE: **Quasi-dynamic network loading: adding queuing and spillback to static traffic assignment.**

ABSTRACT:

For many years, static traffic assignment models have been widely applied in transport planning studies and will continue to be an important tool for strategic policy decisions. As is well known, in the traditional approach, the location of the delays and queues are not predicted correctly, and the resulting travel times do not correspond well with reality. Dynamic models can approach reality much better, but come at a computational cost. In this paper we propose a quasi-dynamic model which inherits most of the computational efficiency of static models, but aims to keep most of the important dynamic features, such as queuing, spillback, and shockwaves. Instead of adjusting the traditional static model or using heuristics, we theoretically derive the model from the dynamic link transmission model, assuming stationary travel demand and instantaneous flow. Furthermore, we present algorithms for solving the model. On a corridor network we illustrate the feasibility and compare it with other approaches, and on a larger network of Amsterdam we discuss the computational efficiency.

KEY WORDS: ***Static traffic assignment; quasi-dynamic traffic assignment; network loading; link transmission model; hard capacity constraints.***

AUTHORS: **Bliemer, Brederode, Wismans & Smits**

CONTACT:

INSTITUTE of TRANSPORT and LOGISTICS STUDIES (C37)
The Australian Key Centre in Transport and Logistics Management

The University of Sydney NSW 2006 Australia

Telephone: +612 9351 0071
Facsimile: +612 9351 0088
E-mail: business.itlsinfo@sydney.edu.au
Internet: http://sydney.edu.au/business/itls

DATE: February 2012

1. Introduction

Traffic assignment models need not much introduction. They are widely used all over the world in transport planning to assist in policy decisions with respect to infrastructure investments. Even though the (academic) research in traffic assignment is shifting towards dynamic traffic assignment, static assignment still remains the most widely used tool in strategic transport planning and will continue to be for some time, as computational efficiency and its simplicity remain important for policy makers. As is known, static models make several very limiting assumptions, resulting in traffic flow predictions that are infeasible (flows exceeding the capacity of the road) and network travel times (important for cost-benefit analyses) that are implausible. In this paper, we propose a new framework for the static network loading problem that takes away most of the current limitations, in particular adding hard capacity constraints and the possibility for traffic flow to spillback to upstream links. Our approach will not take the traditional static traffic assignment model as a starting point, as several others have done, but will take a dynamic traffic assignment model as the base model and derive the static counterpart as a special case. To the best of our knowledge, this is the first time such an approach has been proposed.

Traditional static traffic assignment models consider some given travel time functions (such as the well-known BPR function) in order to determine the link travel times depending on the link flows. Many dynamic traffic assignment models have been proposed that are based on the same concept of travel time functions, depending on the link flows and/or densities at the time of link entrance. These dynamic models have been developed as some kind of dynamic extension to the static traffic assignment problem. The formulation in Janson (1991) is a dynamic version of the traditional static traffic assignment model. Many other DTA model formulations that use travel time functions exist, e.g., Ran et al. (1993), Chen and Hsueh (1998), and Bliemer and Bovy (2003). Also in several dynamic network loading procedures these link travel time functions have also been applied, e.g., see Xu et al. (1999).

Using link travel time functions in a dynamic model causes many problems, as detailed in Bliemer (2007). In general, it becomes impossible to satisfy hard capacity constraints, queues appear in the bottleneck instead of upstream the bottleneck, and spillback is typically neglected. Many traffic flow phenomena cannot be taken into account using travel time functions. Flow propagation should not depend on travel time functions, but rather on traffic flow fundamentals, such that it is consistent with the fundamental diagram.

Therefore, instead of deriving dynamic models from static counterparts, many other dynamic models have been proposed based on theories of traffic flow, such as dynamic models that follow the Lighthill and Whitham (1955), and Richards (1956) model, like the simplified traffic flow theory of Newell (1993), the cell transmission model (Daganzo, 1994, 1995), and the link transmission model (LTM) (Yperman, 2007). These models have shown to be able to more realistically simulate traffic in which shockwaves, queues, and spillback can be accommodated.

Using the knowledge and experience we have with dynamic models, it is now time to revisit the static traffic assignment model. Therefore, in this paper we will derive a static traffic flow model as a special case of a dynamic model. To be more specific, we will use a network extension of Newell's simplified model and derive the static counterpart by making the regular 'static' assumptions, such as considering only a single time period with stationary travel demand. This way, we end up with a static traffic assignment model that will be consistent with traffic flow theory (often captured in a fundamental diagram), explicitly taking capacity constraints into account and accounting for physical queues and spillback. Since such a static model inherits many features from a dynamic model, it is often called *quasi-dynamic*. These quasi-dynamic models do not rely on travel time functions, but rather

on some more realistic traffic flow propagation procedure, from which travel times can be derived afterwards.

4CAST (2009) and Bundschuh et al. (2006) present two of such quasi-dynamic models. These models ensure that link flows do not exceed the link capacity, and determine delays in queues. They describe procedures to determine the flows through a network and the build up of queues at bottlenecks. It should be stressed that only a single time period (with a given stationary flow) is considered in these models, hence they are essentially still static traffic assignment models. Although providing much more realistic results than traditional static models, these models still suffer from a solid theoretical basis, as they are merely presented as algorithms, while the underlying model and assumptions are not made specific. Furthermore, the queues and delays predicted by these models are not consistent with traffic flow theory.

The main contributions of this paper can be stated as follows. First, a quasi-dynamic (static) traffic assignment model is proposed that leads to more realistic outcomes (in terms of link flows, congestion locations, queue lengths, delays, etc.) than any quasi-dynamic (static) model thus far. Secondly, this model has been derived from the dynamic link transmission model, explicitly showing which assumptions have to be made, and providing a solid theoretical basis instead of merely providing a heuristic. Thirdly, we propose an efficient procedure for solving the network loading phase, which is an exact procedure.

The outline of the paper is as follows. Section 2.1 describes the traditional static traffic assignment model and extensions to it in order to include capacity constraints are discussed in Section 2.2. None of these extensions to include capacity constraints seem to lead to realistic results in terms of congestion locations, queues, and spillback. Quasi-dynamic models that have been proposed in the literature are discussed in more detail in Section 2.3. In Section 3 we derive an event-based formulation of the link transmission model, which serves as the starting point for deriving a quasi-dynamic (static) version of this model in Section 4, assuming stationary flow and other 'static' assumptions. In Section 5 we propose an efficient solution procedure for solving our quasi-dynamic traffic assignment problem. Section 6 presents a case study on the network of the city of Amsterdam. Finally, Section 7 closes with a discussion, conclusions, and directions for future research.

2. Static traffic assignment models in the literature

2.1 Traditional static traffic assignment models

In this section, we briefly describe the classical model for static traffic assignment, and introduce notation that we will use in the remainder of the paper.

Consider a transport network $G=(N,A)$ with a set of nodes *N* and a set of directed links, *A*. Let $O \subseteq N$ denote the set of origins, and $D \subseteq N$ the set of destinations. The travel demand between origin-destination (OD) pair (o,d), with $o \in O$ and $d \in D$, is assumed to be given by H^{od}. This travel demand is assumed to be representative for a certain time period $[0,T]$. Let P^{od} be the (non-empty) set of available paths from origin *o* to destination *d*, and define the total route set as $P = \bigcup_{(o,d)} P^{od}$.

Wardrop (1952) defined a user equilibrium in traffic assignment as the situation in which no traveller can be better off (i.e., obtain smaller travel times) by unilaterally switching routes. Traditional static traffic assignment models consider explicit link travel time functions, i.e., continuous monotonously increasing functions $\tau_a(u_a)$ that describe the travel time on link *a*, τ_a, as a function of the flow into link *a*, u_a. Beckmann et al. (1956) showed that Wardrop's user equilibrium flows could be determined by solving the following nonlinear programming problem for the vector of link flows *u*:

$$\min_{u\in\Omega} \sum_a \int_{w=0}^{u_a} \tau_a(w)\,dw, \tag{1}$$

where the set of feasible link flows, Ω, is defined by the following constraints:

$$u_a = \sum_{(o,d)} \sum_p \delta_{ap}^{od} F_p^{od}, \tag{2}$$

$$\sum_p F_p^{od} = H^{od}, \tag{3}$$

$$F_p^{od} \geq 0. \tag{4}$$

The first constraint is a definitional constraint that states that the flow into link *a* is defined by summing all path flows F_p^{od} that pass through link *a*. The route-link incidence indicator δ_{ap}^{od} equals one if link *a* is part of path *p* from *o* to *d*, and is zero otherwise. The second constraint is a flow conservation constraint, which ensures that all travel demand is allocated to one of the available paths. Finally, the third constraint describes the nonnegativity constraints, simply stating that path flows need to be nonnegative. Note that if the path flows are nonnegative, then also the link flows are nonnegative, following from Equation (2).

Several algorithms have been developed to solve for a static user equilibrium, of which the Frank-Wolfe algorithm and the method of successive averages (MSA) are the most well-known ones.

Since in the link travel time functions the travel time merely increases when the link flow exceeds capacity, the capacity is not included as a hard constraint but as a soft constraint. One can therefore not prevent the link flow from exceeding the link capacity. Daganzo (1998) proposed to use a travel time function with an asymptote near the capacity, which aims to prevent the link flow from exceeding the capacity, but cannot guarantee this. Unrealistically high travel times and numerical issues in solving the problem make this choice of travel time function not very popular. Therefore, in the next subsection we will look at traffic assignment problems with explicit (hard) capacity constraints.

2.2 Static traffic assignment models with capacity constraints

Beckmann et al.'s original formulation does not take any explicit link capacity constraints into account. All path flows are assumed to be able to pass through each link, such that the link flows can be determined by a simple mapping from the path flows, given by Equation (2).

In order to take into account that each link *a* has a limited capacity, the following straightforward constraints can be added that defines the set of feasible link flows, Ω,

$$u_a \leq C_a, \quad \forall a. \tag{5}$$

This results in a so-called capacity constrained or what others have defined as the extended Beckmann formulation, and has been investigated by several researchers (e.g., Larson and Patriksson, 1999). Although adding these constraints to the problem seems easy, solving the problem becomes much more tedious, as instead of iteratively solving a shortest path problem, now a multi-commodity minimum cost flow problem needs to be solved (Nie et al., 2004).

Although adding the capacity constraints seems natural, it is not consistent with the link travel time functions $\tau_a(u_a)$, such that 'tricks' with Lagrange multipliers or interior penalty functions are needed. The main problem is, that such travel time functions are not suitable for describing the link flows and link travel times consistently. For example, link travel times depend on the flows on downstream links, as they could block the flow, hence separable travel time functions are not valid. Also note that none of these traditional approaches to capacity constrained assignment result in actual queues. Determining link travel time functions that realistically describe congestion is an almost impossible task. Therefore, other

problem descriptions have been developed that avoid the usage of such functions, as will be described in the next subsection.

2.3 Static traffic assignment models with more realistic traffic propagation

In order to overcome some of the above mentioned problems when using explicit link travel time functions, static traffic assignment models have been proposed that adopt certain flow propagation rules ensuring that the flow does not exceed capacity, such that the travel times are an implicit result.

Bifulco and Crisalli (1998) determine iteratively which number of vehicles of a link can proceed to the next link on their path by checking the corresponding link capacities. Spillback is not taken into account. Nesterov and De Palma (2000) assume lower bounds on the travel time (free-flow travel time) and upper bounds on the flow (link capacity) instead of using link travel time functions. They search for stable traffic equilibria in which the fundamental relationship between flow, speed, and density holds. Queues longer than the link length are assumed not to occur, hence spillback is again not taken into account.

Bundschuh et al. (2006) developed an operation model that they term quasi-dynamic, as it takes capacity constraints and spillback into account, however, at a much smaller computational complexity. They use a kind of incremental assignment (temporarily keeping route choice fixed), in which iteratively a fraction of the travel demand is put on the network, say increases of 5%. The flow is propagated over the consecutive links of a path until the capacity of a link is reached. The extra flow on that link will be stored in the queue of the bottleneck link, and blocked back to upstream links if the queue exceeds the storage capacity of the link. Link travel times are determined afterwards by taking the free-flow travel time and adding delays that refer to the time it takes for the queues to disappear.

4Cast (2009) has developed an operational model called QBLOK, which they also termed quasi-dynamic. This model is a heuristic that ensures that link capacities are not exceeded. The queue starts, similar to the model of Bundschuh et al., in the bottleneck link, and not upstream this link. Queues longer than the link length can occur, such that spillback is taken into account in this model. Since this procedure is a heuristic and not exact, it may produce unrealistic travel times and route choices. The link travel times are derived implicitly from the queues that form.

It is important to note that in traditional static traffic assignment models, we merely talk about link flows, while in these more realistic flow models, we typically talk about link inflows and outflows, as the link inflow and outflow need not be the same due to queues building up on the links. So in the remainder of the paper we will refer to link inflows u_a and link outflows v_a.

2.4 Newly proposed quasi-dynamic traffic assignment model

In this paper we propose a static traffic assignment model in which capacity constraints, spillback, and even shockwaves are explicitly taken into account. This model can be seen as a static version of a network extension of Newell's simplified model, in which a single time period is assumed with a stationary flow. In contrast to Bundschuh et al. and 4Cast, we use traffic flow theory and realistic fundamental diagrams to come to a more rigorous problem formulation. More realistic queuing, including shockwaves, can be taken into account. Heuristics are avoided by computing an exact solution.

In the following sections our quasi-dynamic model will be discussed in more detail, which we will refer to as STAQ (Static Traffic Assignment with Queuing). We will focus on the network loading part and not on the route choice part of the traffic assignment problem, as the route choice part is similar to the traditional approaches. In the next section, we describe the extended Newell model that will serve as the basis for our model derivations.

3. A flow-rate based network extension of Newell's model

This section will describe a network extension of Newell's model. Instead of using cumulative flows as in the original model by Newell and in the LTM model, we will derive a flow-rate based model, as is common in static assignment. Later in Section 4, instead of deriving a time-based algorithm to find an approximate solution as presented in Yperman (2007), our formulation will enable us to formulate an exact event-based algorithm. In this event-based algorithm, the problem will be primarily defined in terms of flow rates (veh/h) rather than cumulative flows (veh) as in Newell (1993) and Yperman (2007). In case of stationary travel demand, an assumption made in static traffic assignment, the link flow rates are (temporarily) constant over time, such that an event-based algorithm will become computationally efficient.

Instead of assuming a fixed, travel demand, we will first consider the more general case of time-varying travel demand between OD pair (o,d), $H^{od}(t)$, and later on make simplifying assumptions.

3.1 Route choice model: Path flows

Let $f_p^{od}(t)$ denote the path flow rate following path $p \in P^{od}$ from origin *o* to destination *d*. This path flow is a result of a route choice model, and depends on the situation one aims to model, e.g., all-or-nothing assignment, deterministic user-equilibrium assignment, stochastic user-equilibrium assignment, or system optimal assignment, etc., and depends on a possible iterative procedure to find this situation, e.g., Frank-Wolfe algorithm, MSA, projected gradient algorithm, etc. Since this paper focuses on the network loading and not on the route choice, we assume that the path choice probabilities are exogenous and given by $\psi_p^{od}(t)$, denoting the probability that at time instant *t* path $p \in P^{od}$ is chosen among all route alternatives in set P^{od}. Hence, we can write

$$f_p^{od}(t) = \psi_p^{od}(t) H^{od}(t). \tag{6}$$

In the following we will omit the indices od, as each path *p* (being a sequence of links) also uniquely defines the origin and destination.

3.2 Definitions

Denote the inflow into and outflow rates out of link $a \in A$ following path $p \in P$ at time instant $t \in [0,T]$ by $u_{ap}(t)$ and $v_{ap}(t)$, respectively.

These path-specific link flow rates can be aggregated by summing over all paths, such that the link inflow and outflow rates are respectively defined as

$$u_a(t) = \sum_p u_{ap}(t), \quad \text{and} \quad v_a(t) = \sum_p v_{ap}(t). \tag{7}$$

The corresponding cumulative inflows $U_a(t)$, which denotes the total number of vehicles that have entered link *a* at time instant *t*, and cumulative outflows $V_a(t)$, denoting the total number of vehicles that have exited link *a* at time instant *t*, are then defined by

$$U_a(t) = \int_0^t u_a(w)\,dw, \quad \text{and} \quad V_a(t) = \int_0^t v_a(w)\,dw, \tag{8}$$

3.3 Flow conservation

The (path-specific) inflow rate into link *a* is given by the (path-specific) outflow rate out of the previous link, say a', on a certain path *p*. In case the link under consideration is the first link on the path, the (path-specific) link inflow rate equals the path flow, $f_p(t)$, as an outcome of the route choice model, see Section 3.1. Therefore,

$$u_{ap}(t)=\begin{cases} f_p(t), & \text{if } a \text{ is the first link on path } p, \\ v_{a^- p}(t), & \text{if } a^- \text{ is the previous link on path } p. \end{cases} \tag{9}$$

3.4 Link model: Sending and receiving flow rates

Following Newell (1993) and Yperman (2007), we assume that the simplified piecewise linear flow-density relationship describes the traffic states on the link, see Figure 1. The functional relationship is given by

$$q_a(k_a)=\begin{cases} \gamma_a k_a, & \text{if } 0 \le k_a \le \dfrac{Q_a}{\gamma_a}, \\ \dfrac{Q_a(K_a-k_a)}{K_a-\dfrac{Q_a}{\gamma_a}}, & \text{if } \dfrac{Q_a}{\gamma_a} < k_a \le K_a, \end{cases} \tag{10}$$

where Q_a, K_a, and γ_a denote the capacity (veh/h), the jam density (veh/km), and the free-flow speed (km/h), respectively. The wave speed ω_a (km/h) as indicated in Figure 1, can be determined from these three (given) quantities, namely $\omega_a = Q_a/(K_a - Q_a/\gamma_a)$.

Figure 1: Newell's simplified fundamental diagram

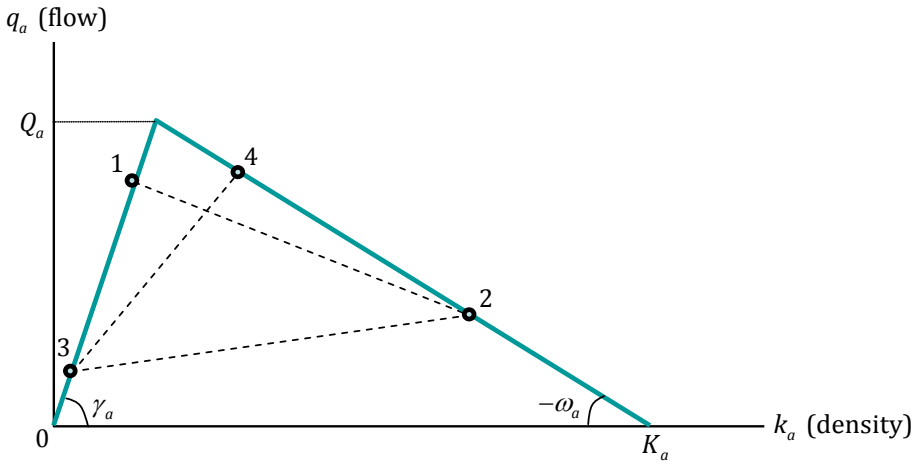


Figure 2: Resulting trajectories and characteristics lines

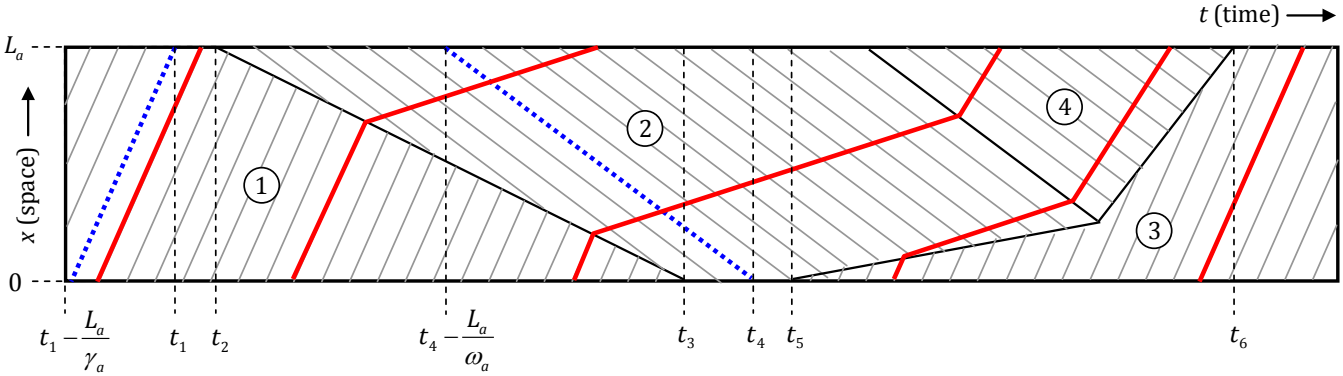


Figure 2 shows the characteristic lines for four different traffic states, corresponding to the traffic states indicated in Figure 1. For illustration purposes, several trajectories through this link have been indicated (in red). Note that the slopes of the shockwaves in Figure 1, which indicate a transition between two different traffic states, are equal to the slopes of the lines connecting the two traffic states in Figure 1. For example, the slope of the line connecting traffic state 1 to traffic state 2 in Figure 1 is the speed of the shockwave between the uncongested state 1and the congested state 2, hence the speed of the tail of the queue.

As derived in Yperman (2007), if at time instant *t* an exiting vehicle traversed the link completely under free-flow conditions, then it holds that

$$U_a\left(t-\frac{L_a}{\gamma_a}\right)-V_a(t)=0. \tag{11}$$

If there is congestion anywhere on the link, this value is greater than zero. By taking the time derivative of Equation (11), we can formulate the following relationship (again assuming free-flow conditions):

$$v_a(t)=u_a\left(t-\frac{L_a}{\gamma_a}\right). \tag{12}$$

Furthermore, if at time instant *t* an entering vehicle is facing a queue that has reached the beginning of the link (e.g., spillback occurs), then it holds that

$$U_a(t)-V_a\left(t-\frac{L_a}{\omega_a}\right)-K_aL_a=0. \tag{13}$$

In other cases, this value is smaller than zero. Taking the time derivative of Equation (13), the following relationship (assuming spillback occurs):

$$u_a(t)=v_a\left(t-\frac{L_a}{\omega_a}\right). \tag{14}$$

The flow rate that potentially could leave link *a* at time instant *t*, also referred to as the sending flow rate, denoted by $s_a(t)$, is equal to $v_a(t)$ given by Equation (12) during free-flow conditions, and is equal to the capacity Q_a in case of congestion (anywhere on the link). Therefore, we can compute the sending flow rates as

$$s_a(t)=\begin{cases} u_a\left(t-\frac{L_a}{\gamma_a}\right), & \text{if } U_a\left(t-\frac{L_a}{\gamma_a}\right)-V_a(t)=0, \\ Q_a, & \text{if } U_a\left(t-\frac{L_a}{\gamma_a}\right)-V_a(t)>0. \end{cases} \tag{15}$$

The flow rate that could enter the link at time instant *t*, also referred to as the receiving flow rate, denoted by $r_a(t)$, is equal to $u_a(t)$ given by Equation (14) during spillback conditions, and is equal to the capacity Q_a in case there is no spillback. Therefore, we can compute the receiving flow rates as

$$r_a(t)=\begin{cases} v_a\left(t-\frac{L_a}{\omega_a}\right), & \text{if } U_a(t)-V_a\left(t-\frac{L_a}{\omega_a}\right)=K_aL_a, \\ Q_a, & \text{if } U_a(t)-V_a\left(t-\frac{L_a}{\omega_a}\right)<K_aL_a. \end{cases} \tag{16}$$

The sending flow rate in Equation (15) equals the inflow rate exactly L_a/γ_a (the free-flow travel time) earlier if there is no congestion, and is equal to the capacity in case of congestion. The receiving flow rate in Equation (16) equals the outflow rate exactly L_a/ω_a (the time for the wave to reach the beginning of the link) earlier if the queue is blocking back, and is equal to the capacity otherwise. As an example, consider again Figure 2. For all $t<t_2$ and $t>t_6$, the sending flow rate is $s_a(t)=u_a(t-L_a/\gamma_a)$, while $s_a(t)=Q_a$ for $t_2<t<t_6$. The receiving flow rate is $r_a(t)=v_a(t-L_a/\omega_a)$ for all $t_3<t<t_5$, while $r_a(t)=Q_a$ for $t<t_3$ and $t>t_5$.

These sending and receiving flow rates are merely potential outflow and inflow rates. The actual flow rates depend also on the upstream and downstream links, as determined by the node model, which will be discussed next.

3.5 Node model: Transition flow rates

The sending and receiving flow rates are flows that would eventuate in case no constraints are put on these flows. However, the sending flow rate is for example constrained by the receiving flow rate of the next link. In general, the sending flow rates will be constrained by intersections where multiple flows come together. A node model is therefore required to determine the actual flow rates in each direction, also referred to as transition flows.

First, we have to decompose the sending flow rates into different directions. The proportion of inflow into link *a* that follows path *p* can be computed as

$$\xi_{ap}(t)=\frac{u_{ap}(t)}{u_a(t)}. \tag{17}$$

The proportion of the inflow rate into link *a* that is travelling in the direction of link *b* is then given by the sum of the path-specific flow rates for which link *b* is the next link,

$$\varphi_{ab}(t)=\sum_p \delta_{bp}\xi_{ap}(t), \tag{18}$$

where δ_{bp} is a link-route incidence indicators that equal one if link *b* is part of route *p*, and zero otherwise. These splitting proportions $\varphi_{ab}(t)$ can also be used to decompose the traffic flow rates at the end of the link. Consider a vehicle that exits link *a* at time instant *t*. Then we can determine the time instant that this vehicle has entered the link, denoted by $\theta_a(t)$, by solving $U_a(\theta_a(t))=V_a(t)$ (since in Newell's model the first-in-first-out condition, FIFO, holds). Therefore, we can apply the splitting proportions $\varphi_{ab}(\theta_a(t))$ to the sending flow rates to obtain the directional sending flow rates, $s_{ab}(t)$, namely,

$$s_{ab}(t)=\varphi_{ab}(\theta_a(t))s_a(t). \tag{19}$$

In this paper we will use the demand-proportional node model proposed in (*6*) to directly determine the transition flows to each of the directions. More sophisticated, often iterative and more time-consuming, node models could be used, see e.g., Tampère et al. (2011). The actual (directional) transition flow rates from link *a* to link *b*, denoted by $g_{ab}(t)$, can be determined from the sending and receiving flow rates as in (see Bliemer, 2007):

$$g_{ab}(t)=s_{ab}(t)\cdot\min\{\zeta_a(t),1\},\ \text{with}\ \zeta_a(t)=\min_{\{b'\in A_a^{\text{out}}|s_{ab'}(t)>0\}}\left\{\frac{Q_{b'}}{\sum_{a'\in A_{b'}^{\text{in}}}s_{a'b'}(t)}\right\}. \tag{20}$$

where A_b^{in} is the set of links upstream of link *b*, and A_a^{out} is the set of links downstream of link *a*.

Using these transition flow rates, the flow rate that actually flows out of link *a* at time instant *t* is

$$v_a(t) = \sum_{b \in A_a^{\text{out}}} g_{ab}(t), \tag{21}$$

and the path-specific link outflow rate is given by

$$v_{ap}(t) = \xi_{ap}\left(\theta_a(t)\right) v_a(t). \tag{22}$$

3.6 Travel times

The link travel time for vehicles entering link *a* at time instant *t*, denoted by $\tau_a(t)$, can due to the FIFO condition easily be derived from the cumulative inflows and outflows by solving

$$U_a(t) = V_a\left(t + \tau_a(t)\right). \tag{23}$$

In other words, we can calculate the link travel times as

$$\tau_a(t) = V_a^{-1}\left(U_a(t)\right) - t. \tag{24}$$

The actually experienced route travel time when departing at time instant *t*, $\tau_p(t)$, can be calculated by recursively adding the appropriate link travel times on the path,

$$\tau_p(t) = \sum_a \delta_{ap} \tau_a\left(\eta_a(t)\right), \text{ with } \eta_a(t) = \begin{cases} t + \tau_{a'}(t), & \text{where } a' \text{ is the previous link on path } p, \\ t, & \text{otherwise.} \end{cases} \tag{25}$$

4. Quasi-dynamic network loading

The flow-rate based dynamic network model as described in the previous section could be efficiently solved using an event-based algorithm if the flow rates do not vary (much) over time. Assuming a stationary travel demand is one way of achieving this, such that in a simple corridor network we can compute the network loading in only a few time steps (events), instead if simulating with small time steps (typically 1 to 5 seconds). However, in a general network with multiple OD pairs, link flow rates change often, since flows from different origins arrive at different time instants at a certain link. Hence, the event-based simulation will quickly become computationally inefficient in realistic networks. Therefore, we make the additional assumption, similar to static assignment models, that traffic flows propagate instantaneously (at least in the free-flowing part of each route) within a single period, ensuring that the link flow rates remain largely stationary. Then the event-based network loading becomes computationally efficient also on large networks.

In this section we derive a quasi-dynamic network loading procedure based on an event-based algorithm to solve the (dynamic) model presented in the previous section, which will be a mix of a static and a dynamic model. Essentially, we make the flow propagation static, but keep parts of the queuing principles dynamic, producing much more realistic traffic flows, queues, and travel times than the traditional static traffic assignment models.

Starting from the dynamic model presented in Section 3, we make two ‘static’ assumptions. The first assumption is, that the travel demand is stationary for a given time period. In other words, we assume an average travel demand for each path, denoted by F_p, for a single time period (we will assume one hour as the basis for the computations; one can reason that in a static model it does not matter what the time period is that one assumes, as long as the travel demand and capacities are consistent with each other). Secondly, we assume that traffic propagates instantaneously through the network, as we essentially consider a single time period, and is only held up by bottlenecks. These two assumptions are consistent with traditional static traffic assignment models, except for the fact that in our model not all traffic will reach its destination at the end of the considered time period, as some traffic may be held up at one or more bottlenecks. In the derivations that follow, we first consider the beginning

of the time period, that is $t = 0,$ and then the remainder of the time period, $t > 0.$ Due to the instantaneous flow propagation assumption, at $t = 0$ traffic flow rates will immediately be propagated through the network, but the link inflow rates cannot exceed the link capacities. Therefore, inflow rates will be capped by the capacities, and the remaining flow rates will again be instantaneously propagated. We call this the *squeezing phase* of the network loading, in which only the flow rates at $t = 0$ are determined. Next, for $t > 0,$ at places where the inflow exceeded the capacity, traffic is held up and will be queued. This will be done in a dynamic fashion, consistent with the link transmission model in which forward and backward shockwaves can occur. We call this the *queuing phase*, in which the cumulative inflows and outflows are determined. The squeezing phase is the static part of the model (for $t = 0$), while the queuing part is the dynamic part of the model (for $t > 0$).

4.1 Squeezing phase ($t = 0$)

First, consider time instant $t = 0.$ Assuming stationary (static) path flows $F_p,$ we can write the flow conservation equation as

$$u_{ap}(0) = \begin{cases} F_p, & \text{if } a \text{ is the first link on path } p, \\ v_{a^- p}(0), & \text{if } a^- \text{ is the previous link on path } p. \end{cases} \tag{26}$$

These path-specific link flow rates can by definition be aggregated to the link flow rates,

$$u_a(0) = \sum_p u_{ap}(0). \tag{27}$$

Since $U_a(0) = 0$ and $V_a(0) = 0$ for all links a, no queues exist yet in the network, so the sending flow rate at $t = 0$ will be equal to the inflow rate. Since we assume instantaneous traffic flow propagation, there is no time lag L_a / γ_a in this inflow rate. The receiving flow rate will be equal to the capacity of the link, as no spillback occurs yet at $t = 0.$ Hence, the sending and receiving flow rates of the event-based link transmission model simplify to

$$s_a(0) = u_a(0), \quad \text{and} \quad r_a(0) = Q_a. \tag{28}$$

Using these sending and receiving flow rates, the transition flow rates in the node model can be simplified to

$$g_{ab}(0) = s_{ab}(0) \cdot \min\{\zeta_a(0), 1\},$$
$$\text{with } \zeta_a(0) = \min_{\{b' \in A_a^{\text{out}} | s_{ab'}(0) > 0\}} \left\{ \frac{Q_{b'}}{\sum_{a' \in A_{b'}^{\text{in}}} s_{a'b'}(0)} \right\}, \text{ and } s_{ab}(0) = \sum_p \delta_{bp} u_{ap}(0). \tag{29}$$

The link outflow rate is the sum of the transition flow rates in any direction out of the link. The path-specific outflow proportions are equal to the path-specific inflow proportions. Hence,

$$v_a(0) = \sum_{b \in A_a^{\text{out}}} g_{ab}(0), \quad \text{and} \quad v_{ap}(t) = \frac{u_{ap}(0)}{u_a(0)} v_a(t). \tag{30}$$

The system of equations (26) – (30) defines the squeezing phase, and results in a solution in which at $t = 0$ all link inflow rates into each link are determined, taking capacity constraints into account. The resulting path-specific link inflow rates $u_{ap}(t),$ obtained through Equation (26), are input into the queuing phase. Note that the above equations are path-based, such that a path-based (iterative) algorithm for solving this system of equations will be proposed in Section 5.1.

4.2 Queuing phase ($t > 0$)

Now let us look at the traffic flows for time instants $t > 0$. Since we assumed instantaneous flow propagation in the squeezing phase, queues will start building up in front of all bottleneck links at $t = 0$. Using the equations in the link and node models from the event-based link transmission model, see Sections 2.4 and 2.5, these queues may spillback to upstream links, and thereby also block outflows of other links, creating forward shockwaves. This may lead to some queues disappearing again over time, as the inflow decreases. This whole dynamic process is kept intact in the quasi-dynamic model we propose here.

The number of events will increase each time a shockwave hits the beginning or end of a link, therefore events may spread out over the entire network. In heavily loaded networks this may lead to an explosion of events being created. Even though the computations remain local for each event, the exponentially increasing number of them may make it computationally infeasible. Therefore, in the algorithm described in the next section we will not add shockwaves when the change in the inflow or outflow rate of a link is very small. The path-based event-based algorithm for solving the queuing phase will be proposed in Section 5.2.

5. Algorithms

In this section we present algorithms for solving the squeezing and queuing phases of the model that were previously described.

5.1 Algorithm for the squeezing phase

In the squeezing phase, we perform a normal network loading according to some given path flows, however, we cut off all path flows that exceed a link capacity. In contrast to other quasi-dynamic traffic assignment models, this algorithm is completely path-based (instead of link based using split proportions), which guarantees that upstream bottlenecks always have the correct effect on downstream flows. A simple algorithm would be to put the first path flow on the network, then the next, etc., and whenever one of the link inflows exceeds capacity, block that link from path flows to enter. However, such an algorithm would be sensitive to the order in which paths are processed, such that path flows from OD pairs that are processed in the beginning could be passing through all links, while the last path flows that are processed may all be blocked. This result is unwanted and therefore we will propose a smarter algorithm in which such an ordering problem does not occur. One way to avoid it is to conduct an incremental assignment, in which in each increment all path flows are considered simultaneously, and within an increment the processing order does not matter since we choose the increments in such a way that no blocking occurs within an increment, only between increments.

While fixed increments of the path flows, say 5%, have been proposed by Bundschuh et al. (1996), this may be both inefficient and inaccurate. It may be inefficient because no link blockings may happen until for example 70% of the flow is assigned, therefore the increment of 5% may be too small. On the other side, it may be inaccurate, as some links may have only a little bit of capacity left, and adding 5% of the path flows may exceed the link capacity, therefore the increment of 5% may be too large. We propose to use varying increments that can be large when possible, and can be small when required. To be more precise, we compute the increments in such a way that the capacity of the link that is closest to capacity will just reach its capacity. Looking back at the previous increment and the impact that had on the link inflow rates, we can predict which link is expected to next reach its capacity, and accordingly adapt the next increment. It may be that in the next iteration the link that was expected to reach its capacity does not reach it due to blocking of path flows through an upstream link that got blocked in the previous increment. Hence, the increment we determine can be too small (loosing efficiency), but is never too large (keeping accuracy). When the link does reach

capacity, for the next increment this link will be termed 'blocked' and no more path flows will be able to enter (and exit), thereby cutting off some path flows.

The first increment is computed somewhat differently. First, all path flows are loaded to the network without any capacity constraints (like a traditional static traffic assignment). Then the link with the highest inflow/capacity ratio is determined, say 1.4, Then the first increment will be 1/1.4 = 71%, which guarantees that no path flows need to be blocked in this first increment.

The algorithm terminates when 100% of all path flows have been loaded onto the network. Due to space limitations we cannot give a detailed description of the algorithm, but a flow chart of the squeezing algorithm is given in Appendix A.

5.2 Algorithm for the queuing phase

Since we are using an event based algorithm for the queuing phase (consistent with the link transmission model), we need to find the time instants in which there is a change in the inflow or outflow rate of a link. For each link we will keep track of their events. We first note that the node model in Equation (20) determines the (directional) link outflow rates depending on sending and receiving flow rates from connecting links, while the link inflow rates are determined from the link outflow rates in a straightforward fashion. Therefore, only when the sending or receiving flow rates change, there will be an event. Thus the algorithm comes down to determining the time instants at which there is such a change. The sending flow rates in Equation (15) only change when either the state of the link changes from free-flow to congestion (or vice versa), or when the (past) inflow rate has changed. The receiving flow rates in Equation (16) only change when the state of the link changes from spillback to no spillback (or vice versa), or when the (past) outflow rate has changed. By just keeping track of these traffic states and changes in inflow and outflow rates, we can exactly compute these time events for each link.

The time instants for the above mentioned four events can quite easily be determined. The simplest ones are the events generated by past changes in inflow and outflow rates. Let $t_{\mathrm{u},a}^{(i-1)}$ and $t_{\mathrm{v},a}^{(i-1)}$ denote the time instants (of the $(i-1)^{th}$ events of link a) at which the inflow and outflow rate changed for the last time, respectively. Then according to Equation (15) the sending flow rate is expected to change at time instant $t_{\mathrm{u},a}^{(i)} = t_{\mathrm{u},a}^{(i-1)} + L_a / \gamma_a$. Similarly, according to Equation (16) the receiving flow rate is expected to change at $t_{\mathrm{v},a}^{(i)} = t_{\mathrm{v},a}^{(i-1)} + L_a / \omega_a$. Note that we use the word 'expected', as this change will occur when no other events before this time occur. If there are other events before these time instants, it may be that the event time has to be adapted, or the event may not even occur at all. Therefore, we keep track of a list of future events (per link) and update them when needed.

Now we will derive the time instants for events in which there is a change of traffic state, which are slightly more complicated. Since $U_a(t)$ is piece-wise linear between time instants $t_{\mathrm{u},a}^{(0)}, t_{\mathrm{u},a}^{(1)}, \ldots, t_{\mathrm{u},a}^{(i-1)}$, we can write

$$U_a(t) = \begin{cases} U_a\left(t_{\mathrm{u},a}^{(j-1)}\right) + u_a\left(t_{\mathrm{u},a}^{(j-1)}\right)\left(t - t_{\mathrm{u},a}^{(j-1)}\right), & \text{if } t_{\mathrm{u},a}^{(j-1)} \le t < t_{\mathrm{u},a}^{(j)}, \text{ for } 1 \le j < i, \\ U_a\left(t_{\mathrm{u},a}^{(i-1)}\right) + u_a\left(t_{\mathrm{u},a}^{(i-1)}\right)\left(t - t_{\mathrm{u},a}^{(i-1)}\right), & \text{if } t \ge t_{\mathrm{u},a}^{(i-1)}. \end{cases} \tag{31}$$

Similarly, it holds that $V_a(t) = V_a\left(t_{\mathrm{v},a}^{(j-1)}\right) + v_a\left(t_{\mathrm{v},a}^{(j-1)}\right)\left(t - t_{\mathrm{v},a}^{(j-1)}\right)$ when $t_{\mathrm{v},a}^{(j-1)} \le t < t_{\mathrm{v},a}^{(j)}$ $(1 \le j < i)$ or $V_a(t) = V_a\left(t_{\mathrm{v},a}^{(i-1)}\right) + v_a\left(t_{\mathrm{v},a}^{(i-1)}\right)\left(t - t_{\mathrm{v},a}^{(i-1)}\right)$ when $t \ge t_{\mathrm{v},a}^{(i-1)}$, where $t_{\mathrm{v},a}^{(i-1)}$ denotes the time instant in which the outflow rate changed for the last time.

Assuming stationary inflow and outflow rates $u_a\left(t_{\mathrm{u},a}^{(i-1)}\right)$ and $v_a\left(t_{\mathrm{v},a}^{(i-1)}\right)$, the time instant $t_{\mathrm{u},a}^{(i)}$ at which the state changes from spillback to no spillback (or vice versa) on link a can be computed by solving $U_a\left(t_{\mathrm{u},a}^{(i)}\right) - V_a\left(t_{\mathrm{u},a}^{(i)} - L_a / \omega_a\right) = K_a L_a$, yielding

$$t_{\mathrm{u},a}^{(i)} = \frac{K_a L_a - U_a\left(t_{\mathrm{u},a}^{(i-1)}\right) + V_a\left(t_{\mathrm{v},a}^{(j-1)}\right) + u_a\left(t_{\mathrm{u},a}^{(i-1)}\right)t_{\mathrm{u},a}^{(i-1)} - v_a\left(t_{\mathrm{v},a}^{(j-1)}\right)\left(t_{\mathrm{v},a}^{(j-1)} + L_a/\omega_a\right)}{u_a\left(t_{\mathrm{u},a}^{(i-1)}\right) - v_a\left(t_{\mathrm{v},a}^{(j-1)}\right)}, \tag{32}$$

where $t_{\mathrm{v},a}^{(j-1)} + L_a/\omega_a \le t_{\mathrm{u},a}^{(i)} < t_{\mathrm{v},a}^{(j)} + L_a/\omega_a$ for $1 \le j \le i$. Furthermore, it needs to hold that $t_{\mathrm{u},a}^{(i)} \ge t_{\mathrm{u},a}^{(i-1)}$. In case this last condition does not hold, the time instant found in Equation (32) can be ignored, as no spillback is expected to occur under the current stationary inflow and outflow rates.

At the downstream side of the link, the time instant $t_{\mathrm{v},a}^{(i-1)}$ at which the state changes from free-flow to congestion (or vice versa) on link a can be computed by solving $U_a\left(t_{\mathrm{v},a}^{(i)} - L_a/\gamma_a\right) - V_a\left(t_{\mathrm{v},a}^{(i)}\right) = 0$, yielding

$$t_{\mathrm{v},a}^{(i)} = \frac{U_a\left(t_{\mathrm{u},a}^{(j-1)}\right) - V_a\left(t_{\mathrm{v},a}^{(i-1)}\right) - u_a\left(t_{\mathrm{u},a}^{(j-1)}\right)\left(t_{\mathrm{u},a}^{(j-1)} + L_a/\gamma_a\right) + v_a\left(t_{\mathrm{v},a}^{(i-1)}\right)t_{\mathrm{v},a}^{(i-1)}}{v_a\left(t_{\mathrm{v},a}^{(i-1)}\right) - u_a\left(t_{\mathrm{u},a}^{(j-1)}\right)}, \tag{33}$$

where $t_{\mathrm{u},a}^{(j-1)} + L_a/\gamma_a \le t_{\mathrm{v},a}^{(i)} < t_{\mathrm{u},a}^{(j)} + L_a/\gamma_a$ for $1 \le j \le i$. Furthermore, it needs to hold that $t_{\mathrm{v},a}^{(i)} \ge t_{\mathrm{v},a}^{(i-1)}$. In case this last condition does not hold, the time instant found in Equation (33) can be ignored, as free-flow conditions are expected to remain under the current stationary inflow and outflow rates.

The time instants stated in Equations (32) and (33) will be added to the event lists for each link, together with the events triggered by changed in inflow and outflow rates. All these (predicted) events will be time-sorted for each link, and the algorithm will work chronologically through these events over all links. The algorithm will terminate at the end of the (one hour) time period. A detailed presentation of the algorithm is beyond the scope of this paper, but for the interested reader an outline of the algorithm is presented in Appendix B, where state I refers to free-flow, state II refers to congestion but no spillback, and state III refers to a congested state with spillback.

5.3 Computation of link travel times

The (average) link travel times can be computed from the cumulative inflow and outflow curves of each link. Observing that the link travel times can be derived from the horizontal distance between the cumulative inflow and outflow curve, the total travel time spent on the link by all vehicles that have entered the link can therefore be determined by the surface between the two curves (which is a straightforward computation, as the cumulative curves are piece-wise linear). This travel time includes the delay experienced by travellers in the queue. When we divide this total travel time by the total number of inflowing vehicles, we obtain the average link travel time.

6. Case studies

In this section we present two case studies. The first case study will be a simple corridor network with multiple bottlenecks, illustrating how the model works. The second case study will be on a realistic network of the city of Amsterdam, in which we focus on computational efficiency.

6.1 Simple corridor network

Let us first consider a simple 7-link corridor network as depicted at the top of Figure 3, in which we have 1, 2, 3, and 4-lane road segments, where we assume each lane has a capacity of 2,000 veh/h, and the travel demand into the corridor is given by an inflow of 4,400 veh/h. Each link is assumed to have the same length (3 km), maximum speed (80 km/h), jam density

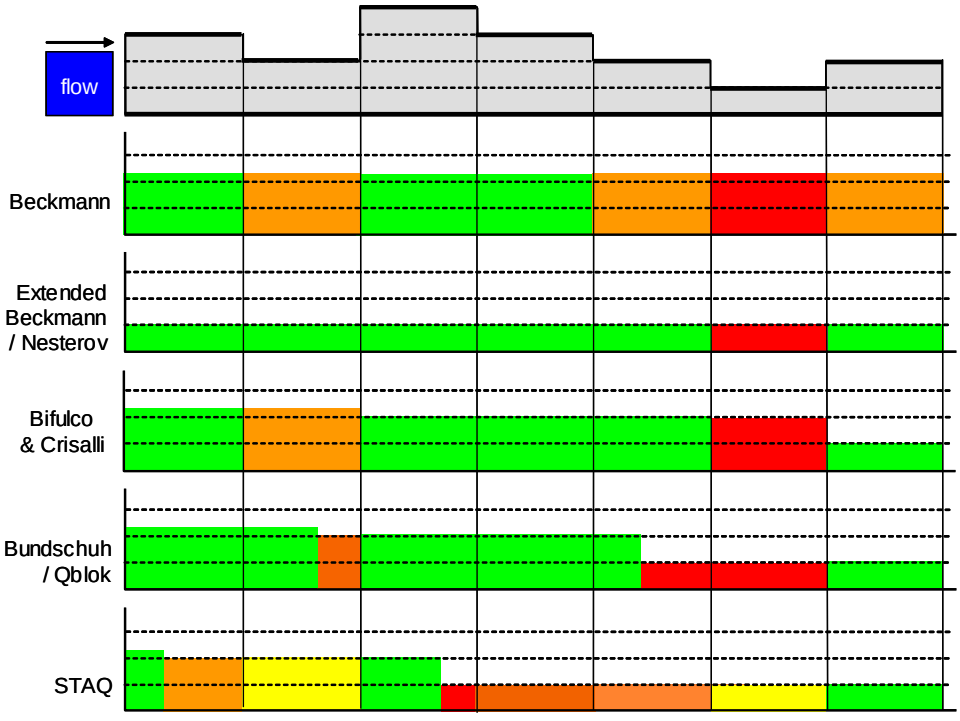
(170 veh/km) and the same wave speed (15 km/h). Given the travel demand and the network supply, we expect a queue to form upstream links 2 and 6.

In the squeezing algorithm, the most critical link will be link 6, with a inflow/capacity ratio of 4,400/2,000 = 2.2, yielding an initial increment of 1/2.2 = 45.5%. Link 6 will be blocked, and the next critical links are links 2 and 5. The second increment will also be 45.5%, blocking links 2 and 5, and the third and final increment will be such that 100% of the travel demand is loaded onto the network.

The link inflow rates from the squeezing phase are then transferred to the queuing phase. Starting at $t = 0$, a queue will build up upstream links 2 and 6. At time $t = 0.195$ (h) the queue on link 5 will spillback to link 4, spilling back further to link 3 at $t = 0.645$. In this case, only 2 event time instants had to be determined. Note that the queuing speed on link 3 is lower than the queuing speed on link 4, which is in turn lower than the queuing speed on link 5. The speed inside the bottlenecks is the speed at capacity. The (average) travel times on link 1 through link 7 are (in hours): 0.089, 0.038, 0.121, 0.268, 0.227, 0.038, and 0.038, respectively, where 0.038 is the free-flow travel time.

The final outcome of our STAQ model is depicted in the bottom picture of Figure 3, where the thickness of the flow shows the inflow rate (at each cross-section), and the colour indicates the speed (green represents high speeds, red low speeds). For comparison, we also show the outcomes of the other models that were discussed in the introduction. Clearly, STAQ provides much more realistic outcomes with respect to the location of the queues, the speeds on the different road segments, and consequently also with respect to the resulting travel times.

Figure 3: STAQ outcome on corridor network and comparison



6.2 Amsterdam network

In order to illustrate an application on a larger more realistic network, we implemented the algorithms in Matlab and used the OmniTRANS toolbox of Matlab to interact with the GUI and databases of OmniTRANS. The travel demand and network supply are taken from a study in which a tactical dynamic model of Amsterdam was built using the dynamic traffic assignment model StreamLine in OmniTRANS. It consists of a total travel demand of almost 100,000 vehicles per hour, and the network has 279 zones, and about 77,000 links, see Figure 4.

We found that in these larger networks, the number of increments in the squeezing algorithm can be quite high (with sometimes extremely small increments) when we would like to have all exact increments, yielding long computation times. In order to decrease the number of increments, we added a minimum increment of 2%. Hence, when possible the exact increment can be large, but when very small increments are computed, we require a minimum of 2% in order to limit the number of increments needed. In this case, we may exceed the link capacity by at maximum 2%, hence the results become only slightly less accurate. This decreased the number of increments in our case by approximately 50%.

In this case study we mainly focus on the computationally efficiency of our STAQ network loading algorithm by comparing it to computation times of a standard network loading algorithm (using BPR functions) within a single (route choice) iteration.

Figure 4: STAQ outcome on Amsterdam network

We also found that the number of events in the queuing phase may increase exponentially each time a forward or backward shockwave (generated by changing link inflow and outflow rates) reaches the end of the link and spreads to the adjacent links. The changes in flow are often very minor, such that we added another rule that skips an event if the inflow or outflow rate of a link does not change more than 1% compared to the previous time the rate was changed. This significantly reduces the number of events while limiting the error in the link flow rates to at maximum 1%.

Further, our new method (with the above mentioned techniques to limit the number of increments and events) seems 8 to 10 times slower than the traditional method (per iteration), mainly due to the squeezing phase and to a lesser extent the queuing phase, as this last phase is performed more locally instead of network wide. Although this is a significant increase in the computation time for a static traffic assignment, it is still much less than running a true dynamic model (with typical time steps of a few seconds), and provides much more realistic traffic patterns. It therefore seems to fill the current gap between static and dynamic models.

It is important to point out that we have not yet investigated the route choice convergence when applied in an iterative procedure. It could be that more iterations are necessary for reaching convergence compared to traditional static assignment. This topic is part of current ongoing research.

7. Conclusions

In this paper we have proposed a new static (or: quasi-dynamic) network loading technique to be used in static traffic assignment, that follows a realistic fundamental diagram and is therefore able to predict more accurate speeds in the queues. The capacity restricts the inflow of the link and therefore queues will correctly build up upstream the bottleneck links. The queuing densities are typically lower than the jam densities considered in the other quasi-dynamic model approaches, such that the queue lengths will be longer.

Comparing our results with the Beckmann and extended Beckmann approach clearly shows that flows following our approach fall in between the two Beckmann approaches. The flows following from Beckmann are too large (not constrained enough), while the flows from extended Beckmann are too small (too much constrained). Further, low speeds and delays occur not in the bottleneck links, but upstream the bottleneck links.

We believe that this new approach is a major improvement over current static network loading models, and fills the gap with the much more elaborate dynamic (simulation based) network loading models. Comparing the computational efficiency with the traditional static procedures, our quasi-dynamic approach requires roughly 10 times more computation time, which is significantly longer, but still much faster than many existing dynamic simulators (taking roughly 100 times more computation time).

Two main research directions have been identified. First, this paper has only dealt with the network loading part and deriving path travel times, and not with the route choice component. The rate of convergence of the traffic assignment with a quasi-dynamic loading is still to be investigated. Secondly, the methodology presented in this paper also opens the door for possibly fast event-based dynamic network loading procedures, which is a subject for future research.

References

4Cast (2009) *Qblok_2004: toedelingsprocedure LMS/NRM*. (in Dutch; working report)

Beckmann, M.J., C.B. McGuire, and C.B. Winsten (1956) *Studies in economics of transportation*, Yale University Press, New Haven CT, USA.

Bifulco, G. and U. Crisalli (1998) Stochastic user equilibrium and link capacity constraints: formulation and theoretical evidences. *Proceedings of the European Transport Conference 1998.*

Bliemer, M.C.J. (2007) Dynamic queuing and spillback in an analytical multiclass dynamic network loading model. *Transportation Research Record 2029*, pp. 14-21.

Bliemer, M.C.J. and P.H.L. Bovy (2003) Quasi-variational inequality formulation of the multiclass dynamic traffic assignment problem. *Transportation Research Part B*, Vol. 37, pp. 501-519.

Bundschuh, M., Vortisch, P. and T. Van Vuuren (2006) Modelling queues in static traffic assignment. *Proceedings of the European Transport Conference 2006*.

Chen, H.-K, and C.-F. Hsueh (1998) A model and an algorithm for the dynamic user-optimal route choice problem. *Transportation Research Part B*, Vol. 32(3), pp. 219-234.

Daganzo, C.F. (1994) The cell transmission model: a dynamic representation of highway traffic consistent with hydrodynamic theory. *Transportation Research Part B*, Vol. 28(4), pp. 269-287.

Daganzo, C.F. (1995) The cell transmission model, Part II: network traffic. *Transportation Research Part B*, Vol. 29(2), pp. 79-93.

Daganzo, C. (1998) Queue spillovers in transportation networks with a route choice. *Transportation Science*, Vol. 32(1), pp. 3-11.

Janson, B.N. (1991) Dynamic traffic assignment for Urban Road Networks. *Transportation Research Part B*, Vol. 25(2/3), pp. 143-161.

Larsson, T. and M. Patriksson (1999) Side constrained traffic equilibrium models – analysis, computation and applications. *Transportation Research Part B*, Vol. 33, pp. 233-264.

Lighthill, M.H., and G.B. Whitham (1955) On kinematics II: a theory of traffic flow on long, crowded roads. *Proceedings of the Royal Society of London*, Series A, No. 299, pp. 317-345.

Nesterov, Y. And A. De Palma (2000) Stable dynamics transportation systems. *Proceedings of the European Transport Conference 2000*.

Newell, G.F. (1993) A simplified theory of kinematic waves in highway traffic, Part I: General theory, Part II: Queuing at freeway bottlenecks, Part III: Multi-destination flows, *Transportation Research Part* B, Vol. 27, pp. 281-313.

Nie, Y., Zhang, H. and D. Lee (2004) Models and algorithms for the traffic assignment problem with link capacity constraints. *Transportation Research Part B*, Vol. 38, pp. 285-312.

Ran, B., D.E. Boyce, and L.J. LeBlanc (1993) A new class of instantaneous dynamic user-optimal traffic assignment models. *Operations Research*, Vol. 41(1), pp. 192-202.

Richards, P.I. (1956) Shockwaves on the highway. *Operations Research*, Vol. 4, pp. 42-51.

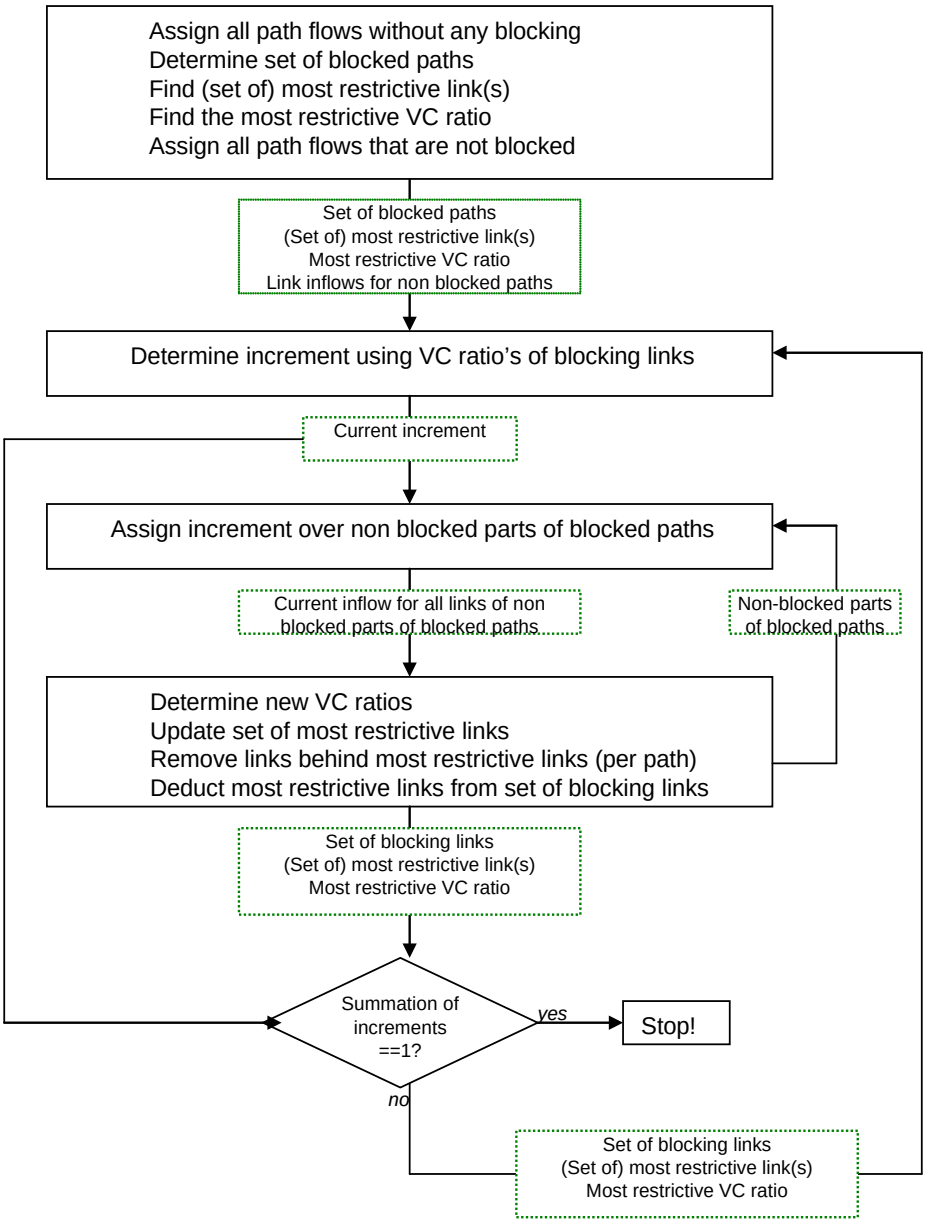
Tampère, C.M.J. , Corthout, R., D. Cattrysse and L.H. Immers (2011) A generic class of first order node models for dynamic macroscopic simulation of traffic flows, *Transportation Research Part B*, Vol. 45, pp. 289-309.

Wardrop, J.G. (1952) Some theoretical aspects of road traffic reseach. *Proceedings of the Institute of Civil Engineers*, Part II, pp. 325-378.

Xu, Y.W., H. Wu, M. Florian, P. Marcotte, and D.L. Zhu (1999) Advances in the continuous dynamic network loading problem. *Transportation Science*, Vol. 33(4), pp. 341-353.

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

Appendix A: Outline of the squeezing algorithm



Appendix B: Outline of the queuing algorithm