STAQ: STATIC TRAFFIC ASSIGNMENT WITH QUEUING

Luuk Brederode
Goudappel Coffeng BV, The Netherlands
Michiel Bliemer
Goudappel Coffeng BV & Delft University of Technology, The Netherlands
Luc Wismans
Goudappel Coffeng BV & University of Twente, The Netherlands

ABSTRACT

Because of computation time issues on large networks, most strategic regional and urban transport models today use static instead of dynamic traffic assignment procedures. Mathematical models of traffic assignment are usually based upon Wardrop's principle. To solve this static traffic equilibrium problem almost all applied static assignment models follow Beckmann who formulated it as a convex optimization problem containing a link travel time function (Beckmann 1956). This function has the form of a polynomial whose degree and coefficients are specified from statistical analysis of real data. The best known polynomial is the BPR function (US bureau of Public Roads, 1964). Although widely used, traffic assignment models based on Beckmann's formulation have several drawbacks. Firstly, these models penalize but not explicitly constrain link flows to their respective link capacities. This can result in a solution where traffic flows exceed link capacities. Secondly, models derived from Beckmann's formulation do not account for queuing and spillback on the network as a result of high demand, resulting in poor travel times and route choice on congested networks. Related drawbacks are that congestion is modelled downstream instead of upstream from the bottleneck and that upstream bottlenecks do not influence downstream traffic demand. These drawbacks not only yield incorrect link flows and travel times, they also prevent proper network and matrix calibration using traffic counts on congested links. Given the ever increasing levels of structural congestion, these drawbacks will only become more relevant in the future.

In order to overcome the drawbacks of Beckmann's formulation of the static traffic equilibrium problem, models that explicitly constrain link flows to their respective capacities have been proposed, both in literature (e.g. Marcotte et al (2003), De Palma and Nesterov (2000), Larsson and Patriksson (1999), Bifulco and Chrisali (1998)), and practice (e.g. Bundschuh et al (2006), 4Cast (2009)). All these models cope with queuing thereby producing more accurate travel times and placing congestion upstream from the bottleneck. However, these models still have drawbacks, such as using heuristic or approximate traffic delay rules and/or lack spillback-effects.

Given these drawbacks and the ever increasing levels of structural congestion, there is a great need for a static traffic assignment model which can be applied on both regional and urban regions, taking both queuing as well as spillback into account. Instead of improving existing traffic assignment models, we propose to start with a dynamic assignment model and construct

the special case of a static traffic assignment model. This leads to STAQ (Static Traffic Assignment with Queuing), a specific case of the link transmission model (Yperman 2007), with the added assumptions that there is only one time period in which there is a stationary travel demand. While the iterative (equilibrium) route choice model is the same as existing methods (e.g., solving the deterministic or stochastic equilibrium problem with MSA or Frank-Wolfe), STAQ replaces the traffic flow and travel time computations (basically the static equivalent of dynamic traffic propagation) and is event-based. STAQ exhibits many favourable properties of dynamic models (such as horizontal dynamic queuing, shockwaves, spillback, using the complete fundamental diagram) but simplified such that it is suitable for strategic planning studies on large scale networks.

In this paper the concept of STAQ is explained and discussed, as well as some case studies on some smaller hypothetical networks and a real life network.

1. INTRODUCTION

Because of computation time issues on large networks, most strategic regional and urban transport models today use static instead of dynamic traffic assignment procedures. These models however, cannot cope with blocking back and spillback of traffic. These drawbacks becomes have become more important over the years given the use of transport models. Driven by e.g. cost-benefit analysis and spatial accessibility studies, their field of use has shifted from solely forecasting traffic volumes on networks with relatively little detail towards forecasting both traffic volumes as well as travel times on networks with much more detail. This 'abuse' of static assignment models has lead to poor results and (legitimate) doubt about the use and even right of existence of transport models as a whole. Therefore, in this paper we propose a new method which combines the advantages of static (its speed and scalability) and dynamic (its realistic flow propagation accurate travel times) assignment models. This new static traffic assignment model with queuing follows traffic flow theory and is based on realistic fundamental diagrams

2. STATIC TRAFFIC ASSIGNMENT

In this section, first the traditional static traffic assignment model is described. Then the extension to include capacity constraints is described and several solution approaches proposed in literature are discussed. Since these traditional (extended) static traffic assignment models do not produce very realistic traffic flows, other approaches have been proposed, which will be briefly discussed. These more realistic approaches make it possible to model queues and spillback, but still have several drawbacks. Then the new static traffic assignment model with queuing is introduced. Finally, a comparison between the different approaches is illustrated in a small example.

2.1 Traditional static traffic assignment without capacity constraints

Consider a transportation network $G=(N,A)$, consisting of nodes N and links A. Denote the set of origin nodes by $R \subseteq N$ and the set of destination nodes by $S \subseteq N$. Let P^{rs} denote the set of paths from origin $r \in R$ to destination $s \in S$. Furthermore, let D^{rs} be the given travel demand from origin r to destination s. Then the set of feasible link flows, Q, can be defined by the following constraints:

$$\sum_p f_p^{rs} = D^{rs}, \quad \forall (r,s), \tag{1}$$

$$q_a = \sum_{(r,s)} \sum_p \delta_{ap}^{rs} f_p^{rs}, \quad \forall a, \tag{2}$$

$$f_p^{rs} \geq 0, \quad \forall (r,s), \forall p, \tag{3}$$

where f_p^{rs} is the path flow on path $p \in P^{rs}$ from r to s, q_a is the link flow on link $a \in A$, and δ_{ap}^{rs} is a route-link incidence indicator that equals one if link a

is on path p from r to s, and equals zero otherwise. Equation (1) is a flow conservation constraint that indicates that the travel demand should be satisfied. Equation (2) is a definitional constraint that states that the link flows are composed of all the path flows that go through that link. Finally, inequality (3) is a nonnegativity constraint, ensuring that all flows are nonnegative.

All link flows $q \in Q$ are feasible, but we are interested in finding a user equilibrium solution that satisfies Wardrop's equilibrium law (Wardrop, 1952). Wardrop's first principle states that for each origin-destination (OD) pair, all used paths have equal travel time, and there exist no unused paths with a lower travel time. In other words, if $f_p^{rs} > 0$, the path travel time τ_p^{rs} is equal to the minimum over all available paths in P^{rs}. If $f_p^{rs} = 0$, the path travel time may be larger than this minimum. Let F be the set of feasible path flows that satisfy constraints (1) and (3). It can be shown (see e.g., Nagurney, 1993) that the solution to the following variational inequality (VI) problem describes a Wardrop user equilibrium. The VI problem is to find path flows $\bar{f} \in F$ such that

$$\sum_{(r,s)} \sum_{p} \tau_p^{rs}(\bar{f}) \left(f_p^{rs} - \bar{f}_p^{rs} \right) \geq 0, \quad \forall f \in F, \tag{4}$$

where the path travel times are defined as $\tau_p^{rs} = \sum_a \delta_{ap}^{rs} \tau_a$, and where F is the set of feasible path flows that satisfy constraints (1) and (3). Using the definition of the path travel time and constraints (2), we can rewrite this path-based VI problem into a link-based VI problem:

$$\begin{aligned} \sum_{(r,s)} \sum_{p} \tau_p^{rs}(\bar{f}) \left(f_p^{rs} - \bar{f}_p^{rs} \right) &= \sum_{(r,s)} \sum_{p} \sum_{a} \delta_{ap}^{rs} \tau_a(\bar{q}) \left(f_p^{rs} - \bar{f}_p^{rs} \right) \\ &= \sum_{a} \tau_a(\bar{q}) \sum_{(r,s)} \sum_{p} \delta_{ap}^{rs} \left(f_p^{rs} - \bar{f}_p^{rs} \right) \\ &= \sum_{a} \tau_a(\bar{q}) \left(q_a - \bar{q}_a \right) \end{aligned} \tag{5}$$

Hence, in the link-based VI problem we would like to find link flows $\bar{q} \in Q$ such that

$$\sum_{a} \tau_a(\bar{q}) \left(q_a - \bar{q}_a \right) \geq 0, \quad \forall q \in Q. \tag{6}$$

Since a VI problem of finding an $\bar{x} \in X$ for which $g(\bar{x})^T (x - \bar{x}) \geq 0, \forall x \in X$, can be rewritten as an equivalent optimization problem of the form $\min_{x \in X} h(x)$, with $g(x) = \nabla_x h(x)$ (under the condition that $\nabla_x g(x)$ is a symmetric matrix), VI problem (6) can be rewritten as the following optimization problem:

$$\min_{q \in Q} \sum_{a} \int_{\omega=0}^{q_a} \tau_a(\omega) d\omega. \tag{7}$$

This optimization problem corresponds to the well-known formulation by Beckmann et al. (1956), where each link travel time only depends on the flow on that link. In other words, the travel time functions are assumed to be separable. If these travel time functions are also continuous and non-decreasing, the optimization problem is convex and has a solution. If the functions are strictly increasing, the solution is unique (Smith, 1979). In practice, the Bureau of Public Roads (BPR) travel time functions are often used,

$$\tau_a(q_a) = \tau_a^0 \left(1 + \alpha_a \left(\frac{q_a}{C_a} \right)^{\beta_a} \right), \tag{8}$$

where C_a is the capacity of link a, τ_a^0 is the free-flow travel time, and α_a and β_a are some given link parameters.

Several algorithms have been developed to solve problem (7), of which the Frank-Wolfe algorithm is the most well-known, which iteratively solves a shortest path problem to determine the steepest descent and calculates an optimized step size. An often used heuristic is the method of successive averages in which the stepsize is equal to 1/n where n equals the number of iterations.

Since the link travel time merely increases when the link flow exceeds capacity, capacity is not included as a hard constraint but as a soft constraint. One can therefore not prevent the link flow to exceed the link capacity. Daganzo (1977a,b) 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 section we will look at traffic assignment problems with explicit (hard) capacity constraints.

2.2 Traditional static traffic assignment 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, Q,

$$q_a \leq C_a, \quad \forall a. \tag{9}$$

This results in a so-called capacity constrained or extended Beckmann formulation. Although adding these constraints to the problem is 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).

The capacitated problem can be solved by a sequence of uncapacitated problems using either an exterior penalty function (also called the augmented Lagrangean method), or by means of interior penalty functions.

To reveal the concept of external penalty functions, the extended Beckmann problem can be rewritten in terms of Karush-Kuhn-Tucker (KKT) conditions, in which Lagrange multipliers are associated with the constraints. Suppose that λ_a is the Lagrange multiplier associated with capacity constraints (8). Solving the KKT conditions yields a solution in which $\lambda_a = 0$ if $q_a \leq C_a$, and $\lambda_a > 0$ if $q_a = C_a$ (i.e., the constraint is binding). The Lagrange multiplier is therefore often interpreted as the extra cost or delay on link a on top of the link travel time τ_a, see Yang and Yagar (1994, 1995) and Larsson and Patriksson (1995).

Interior penalty functions try to approximate the constrained traffic assignment problem by adding a penalty term to the objective function of the unconstrained problem, see Nie et al. (2004) and Prashker and Toledo (2004). Shahpar et al. (2008) describe a new solution method in which the side constraints are taken into consideration by implicitly adding a penalty function to the link travel times, which they call the dynamic penalty function method. In several tests they show that this new method achieves faster convergence to a solution than the augmented Lagrangean method or the inner penalty function approach.

Although adding the capacity constraints seems natural, it is not consistent with the link travel time functions $\tau_a(q_a)$, 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 should be developed that avoid the usage of such functions, as will be described in the next section.

2.3 More realistic static traffic assignment with capacity constraints

In dynamic traffic assignment it has been argued that models using link travel time (performance) functions, see e.g. Janson (1991), Ran and Boyce (1996), Chen and Hsueh (1998), and Bliemer and Bovy (2000), cannot realistically describe the traffic dynamics, such as queuing and spillback. Instead, flow propagation models that explicitly follow the traffic flow theory, with fundamental diagrams as input, have been successfully employed in simulation-based assignment, see e.g. the cell transmission model (Daganzo, 1994) or the link transmission model (Yperman, 2007). The first dynamic traffic assignment models were basically dynamic extensions of the static assignment problem, in which the link travel time functions became functions of the link inflows or volumes at a certain time instant. Here we will describe some static traffic assignment models that have been proposed in the literature that aim to take more realistic queuing into account. All these approaches solve iteratively a shortest path problem to determine the user equilibrium which is the same as the traditional approaches do. These approaches however, differ in the way traffic is loaded on the network and/or travel times are determined.

Bifulco and Crisalli (1998) determine iteratively the number of vehicles on a link that can proceed to the next link on their path by checking the corresponding link capacities. This means that not all traffic will be able to reach its destination in the time interval considered. Spillback is not taken into account.

Nesterov (2000), and Nesterov and De Palma (2000a,b, 2001) 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. In order to determine travel times, they use incremental loading of the network, 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, 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. The calculation of travel time in this model is done using a heuristic that ensures that link capacities are not exceeded, and queues appear upstream of bottleneck links. Queues longer than the link

length can occur, taking blocking back into account. Since this procedure is a heuristic, it may produce unrealistic travel times and route choices. Also, the actual link flows that QBLOK produces can be greater than link capacities because these are calculated using a classical static assignment procedure based on the travel times produced by the QBLOK heuristic.

In all of the above approaches, the link and path travel times are not explicit functions, but are derived implicitly from the queues that form.

2.4 Newly proposed static traffic assignment with capacity constraints

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 the link transmission model (Yperman 2007), in which a single time period is assumed with a stationary traffic demand. It somewhat resembles the models proposed by Bundschuh et al. (2006) and 4Cast (2009), and can therefore also be called quasi-dynamic. In contrast to Bundschuh et al. (2006) and 4Cast (2009), we use traffic flow theory and realistic fundamental diagrams to come to a more rigorous problem formulation. More realistic queuing, including shockwaves, is taken into account. Heuristics are avoided by computing an exact solution. The model proposed in this paper will be referred to as STAQ: Static Traffic Assignment with Queuing. Similar to the above mentioned approaches, link and path travel times are computed implicitly after determining the flows and the queues. The path travel times are used in an iterative route choice scheme in order to solve VI problem (4), where the feasible flows F obey traffic flow theory and the fundamental diagrams.

2.5 Comparison of approaches

To illustrate the differences in the link flows and speeds between the approaches discussed above, consider the simple corridor network in Figure 1. The path consists of seven links with varying link capacities, ranging from a one lane to a four lane motorway segment. The flow (veh/h) through this corridor is assumed to be somewhat larger than the capacity (veh/h) of a two lane segment. In real life, this would result in a queue building up upstream of the second link and upstream of the sixth link. The link flows and the corresponding speeds (indicated by the colour of the flow, see Figure 2 in the appendix for an explanation of the colours) for the original Beckmann approach, the extended Beckmann approach, the approach by Bifulco and Crisalli, the approach by Bundschuh et al., and our newly proposed approach, are indicated in the figure. Note that any delay penalties have been incorporated in the link speed colours.

The original Beckmann formulation without capacity constraints clearly does not restrict the link flow to the link capacity, such that unrealistic flows appear.

Low speeds (corresponding to high travel times) are predicted in the bottleneck links.

The extended Beckmann model with capacity constraints ensures that no link flows exceed the link capacity. However, the result is too restrictive, as the whole path flow is now restricted to the most critical link capacity, while the flow through the first five links only needs to be restricted to the two lane capacity. A (delay) penalty is put on the most restrictive bottleneck link.

Bifulco and Crisalli achieve a more realistic traffic flow pattern after putting the entire travel demand on the path. Since they assume that the capacity restricts the outflow capacity of a link and not the inflow capacity, queues build up inside the bottleneck links, not upstream of the bottleneck links. Spillback can not occur and speeds do not follow a given fundamental diagram, but are based on link cost functions.

Bundschuh et al. yields a similar traffic flow pattern when compared to Bifulco and Crisalli. Queues build up inside bottleneck links, but they may spillback to upstream links. The speeds in bottleneck links decrease because of the queues and may also decrease the speeds on upstream links if spillback occurs. Speeds do not follow a given fundamental diagram, but are based on simple queuing theory.

Our newly proposed model, STAQ, follows a realistic fundamental diagram (see appendix) 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 speeds are typically larger than in the Bundschuh et al. model, and the queue lengths, determined by the speeds of the shockwaves, are therefore also larger. The speed inside the bottleneck is the speed at capacity, which may be smaller than the free-flow speed.

Comparing the STAQ results with the Beckmann and extended Beckmann approach clearly shows that flows following the STAQ procedure 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.

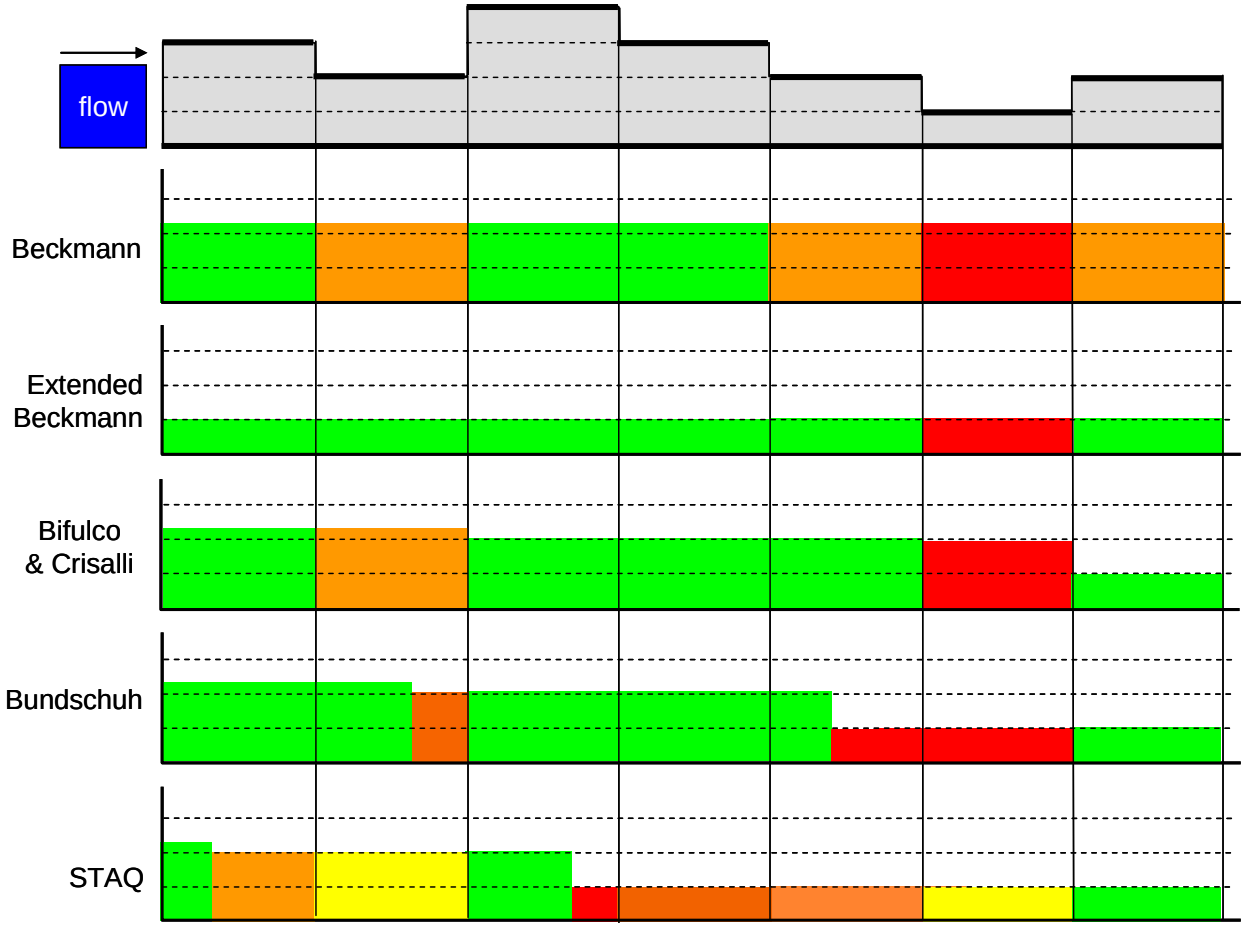

Figure 1: Example corridor network and resulting flows for different approaches

3. STAQ: STATIC TRAFFIC ASSIGNMENT WITH QUEUING

In this section the STAQ model is described in detail. First, the outline of the algorithm is described. Then the problem formulation and an solution algorithm are described for both phases that form STAQ.

3.1 Outline of STAQ

STAQ is a flow propagation model: it merely computes travel times based on traffic flows derived from a given traffic demand for a given study period. Route choice is considered exogenous to the model and can be solved using any Frank-Wolfe type algorithm.

STAQ consists of two phases: the squeezing phase and the queuing phase. After the queuing phase travel times can be derived from the network.

In the first phase (squeezing) traffic demand from all origin-destination pairs is put on the network along paths derived from an earlier performed route choice model. When traffic demand on a link is greater than its capacity this link is considered a bottleneck link. Traffic flow over all paths using one or more

bottleneck links is reduced ('squeezed') from the bottleneck link(s) to the destination. The surplus of traffic flow is stored as a vertical queue at the start of the bottleneck link(s). The squeezing phase yields inflows (the amount of traffic that flows into a link) and outflows (the amount of traffic that flows out of a link) for all links that form the network consistent with the available capacity and route choice of traffic. Note that after the squeezing phase already more realistic travel times can be computed based on the determined vertical queues at the bottlenecks. However, to be able to take the effects of spill back into account the algorithm proceeds with a second phase.

In the second phase (queuing), the vertical queues are translated into horizontal queues using traffic flow theory and fundamental diagrams. In STAQ, a queue is explicitly modelled as a shockwave which is propagated through the network. Shockwaves mark a change in flow conditions (density, flow and speed) over space and can merge, split and cause new shockwaves when propagated through the network. An extensive description of the fundamental diagram used in STAQ as well as the way it is used can be found in the appendix. The queuing phase yields flow conditions on any given location on any given link on the network. Based on cumulative flows, travel times and link speeds can be derived.

3.2 STAQ: Squeezing phase

The squeezing phase determines the amount of traffic that flows into each link, taking into account the link capacities, but no blocking back and spillback can occur. The solution algorithm therefore assigns increments of the total flow over all paths and calculates the increments in such a way that none of the increments results in path flows where flow over a link exceeds capacity. This way the algorithm outcomes are not dependent of the order in which the different paths are assigned to the network. By making the size and thus the number of increments dependent of the traffic demand and link capacities, computation time of the squeezing algorithm also depends of these variables.

Unlike fully dynamic assignment models, the squeezing algorithm in STAQ has no time variable. The time dependent variables (traffic flow and link capacity) are made time-independent by assuming a stationary traffic demand over a given time period (the study period as defined by the researcher). This assumption creates implications on the study period when applying the queuing algorithm. These implications will be discussed in section 5.

Squeezing phase: problem formulation

The problem of the squeezing phase can be described as the system of equations (10), (11), (12) and (13) (derived from Yperman (2007)):

$$q_{ap}^{in} = q_{a^- p}^{out} \qquad \forall a \in A, a^- \in A, p \in P \tag{10}$$

$$q_{ap}^{out} = q_{ap}^{in} * \frac{G_{ab}}{S_{ab}} \qquad \forall a \in A, b \in A, p \in P \tag{11}$$

In which q_{ap}^{out} is the outflow of a link caused by traffic on path *p*, a^- the preceding link of *a* on path *p* and b the succeeding link of *a* on path *p*.

Equation (10) connects the different links on a path by setting the inflow of a link equal to the outflow of a preceding link. Equation (11) reduces the outflow of a link to reflect restrictions on links connected to the downstream end of the link. The amount of reduction is equal to the ratio between the actual possible flow (equation (12)) and the amount of traffic wanting to flow from the considered towards the next link on path *p* (equation (13)).

$$G_{ab} = \min_{b'}\left\{\frac{C_{b'}}{\sum_{a'\in A} S_{a'b'}} * S_{ab}; S_{ab}\right\} \quad \forall a \in A, b \in A \tag{12}$$

$$S_{ab'} = \sum_{p} \delta_{ap} * q_{ap}^{in} \qquad \forall a \in A, b' \in A \tag{13}$$

Where a' is the preceding link of link *b* on any path and b' the succeeding link of *a* on any path.

Squeezing phase: algorithm
Below the squeezing algorithm as currently implemented in a prototype is described.

Step1: Initialization

- Let *P* be the set of paths. $P = \{P^{rs} \mid r \in R, s \in S\}$
- Assign all path flows without any blocking, and compute the link flows using equation (x2)
- Determine all link V/C ratios, $\xi_a = q_a^{in} / C_a$
- Determine the set of blocking links, $\bar{A}$, $\bar{A} = \{a \in A \mid \xi_a > 1\}$.
- Determine the set of blocked paths, $\bar{P}$, $\bar{P} = \{p \in P \mid \exists a \in \bar{A} : \delta_{ap} = 1\}$.
- Find the current most restrictive link, $\bar{a}$, in which $\xi_{\bar{a}}^{\text{ini}} = \max_{a\in\bar{A}}\{\xi_a\}$.
- Set $i := 0$.
- Set $q_a^{in,0} = \sum_{p\in P\setminus\bar{P}} \delta_{ap} f_p$ (assign all path flows that are not blocked)
- Set $q_a^{out,0} = q_a^{in,0}$ for all path flows that are not blocked
- Set turn flows $tf_{ab} = q_a^{out,0}$ for all turns on paths that are not blocked
- Set $\xi_a^{(i)} = 0, \quad \forall a \in \bar{A}$.

Step 2: Assign increment

- Determine increment: $\theta^{(i+1)} = \begin{cases} \dfrac{1}{\xi_{\bar{a}}^{\text{ini}}}, & \text{if } i = 0, \\ \theta^{(i)} \cdot \min\limits_{a\in\bar{A}}\left\{\dfrac{1-\xi_a^{(i)}}{\xi_a^{(i)} - \xi_a^{(i-1)}}\right\}, & \text{if } i > 0. \end{cases}$

- Set $q_a^{in,(i+1)} = q_a^{in,(i)} + \sum_{p \in \bar{P}} \delta_{ap} f_p \theta^{(i+1)}$
- Set $q_a^{out,(i+1)} = \begin{cases} q_a^{in,(i+1)} & \forall a \in \bar{P} \\ q_a^{out,(i)} & \forall a \notin \bar{P} \end{cases}$
- Update turn flows $tf_{ab} := tf_{ab} + q_b^{in(i+1)} \qquad \forall ab \mid b \in \bar{P}$
- Set $i := i+1$.
- Update $\xi_a^{(i)} = q_a^{in,(i)} / C_a, \qquad \forall a \in \bar{A}$ [Note that all V/C ratios are below or equal to one].
- Update the set of actively blocking links, $A^* = \left\{ a \in \bar{A} \mid \xi_a^{(i)} = 1 \right\}$.
- For all actively blocking links $a \in A^*$, update the link sequences in path set $\bar{P}$ such that the paths only contain links up to the first actively blocking link.
- Update the set of blocking links, $\bar{A} := \bar{A} \setminus A^*$.

Step 3: Convergence

- If $\sum_i \theta^{(i)} = 1$, then stop. Otherwise, return to Step 2.

3.3 STAQ: queuing phase

The queuing phase calculates the effect of blocking back and spillback given the inflows and vertical queues calculated in the squeezing phase. In order to do so the algorithm keeps track of the speed of and flow conditions around all shockwaves travelling through the network. When any shockwave reaches the beginning or end of a link, or reaches another shockwave, a so called event occurs. Because the speed of each shockwave is known, it is possible to determine the point of time (relative to the start of the queuing algorithm t0) on which the event occurs. Flow conditions around the original shockwave are adjusted taking into account the existing link and flow conditions on the connecting link(s).

The queuing phase ends when all traffic demand has reached its destination. Because in STAQ a single time period and stationary flow is assumed all traffic demand is ‘put on the network’ when the time indicator in the queuing phase equals the length of the study period. This does not mean that all traffic demand has reached its destination at the end of the study period because traffic can be held up at bottlenecks. The queuing phase therefore continues after the time indicator equals the length of the study period, but with inflows on all links set to 0 (creating new forward shockwaves), reflecting that all traffic demand has been put on the network. The queuing phase ends when no more shockwaves are present, i.e.: all traffic demand has reached its destination.

Queuing phase: problem formulation

Let $(q_a^{\text{in}}, k_a^{\text{in}})$ and $(q_a^{\text{out}}, k_a^{\text{out}})$ denote the traffic conditions at the beginning and at the end of link a, respectively. Transitions between two traffic states are given by a shockwave that can move backward or forward through a link. Let $\left(q_a^{(n)}, k_a^{(n)}\right)$ describe the traffic conditions downstream of the n^{th} shockwave, counted from the beginning of the link. If for example the outflow rate is smaller than the inflow rate, possibly because of a downstream bottleneck, there will exist a backward shockwave. Any changes in the outflow capacity of a link may again start such a backward shockwave. If the inflow conditions change, this may yield a forward moving shockwave. Hence, several shockwaves may exist on a link at a certain time instant. In case these shockwaves meet each other, a new shockwave is formed with a new speed and direction (forward or backward), depending on the traffic states upstream and downstream the shockwave. Shockwaves have a speed that is given by

$$w_a^{(n)} = \frac{q_a^{(n)} - q_a^{(n-1)}}{k_a^{(n)} - k_a^{(n-1)}}. \tag{14}$$

where $\left(q_a^{(n)}, k_a^{(n)}\right)$ is the traffic state downstream the shockwave, and $\left(q_a^{(n-1)}, k_a^{(n-1)}\right)$ is the traffic state upstream. In case $w_a^{(n)} < 0$, the shockwave will move backward, while $w_a^{(n)} > 0$ indicates a forward moving shockwave. Let $x_a^{(n)}(t_a^{(n)})$ be the location of shockwave n that started at time instant $t_a^{(n)}$. The location of this shockwave at time instant t is given by

$$x_a^{(n)}(t) = x_a^{(n)}(t_a^{(n)}) + w_a^{(n)}\left(t - t_a^{(n)}\right). \tag{15}$$

There are three situations in which new shockwaves are created. Either at the beginning of a link due to a change in the inflow conditions, such that $x_a^{(n)}(t_a^{(n)}) = 0$, at the end of a link due to changes in the outflow capacity, such that $x_a^{(n)}(t_a^{(n)}) = L_a$, or somewhere else on the link when two shockwaves meet each other, with $0 < x_a^{(n)}(t_a^{(n)}) < L_a$. To illustrate, consider Figure 2. At t_0, we assume that the outflow capacity drops, such that a queue will form. The tail of the queue will move backward with a shockwave speed equal to the slope from traffic state 2 to traffic state 1 indicated in the fundamental diagram. At time t_1, assume that the inflow rate into the link drops, changing the traffic state to 3. A forward moving shockwave will result with a speed equal to the slope from traffic state 3 to traffic state 1 in the fundamental diagram. At time t_2, we assume another drop in the outflow capacity, changing the outflow conditions to traffic state 4, and another backward shockwave is created. As can be seen at t_3, the first and second shockwave (counted from the left) approach each other, while the third shockwave may overtake the second shockwave due to its higher speed. At t_4, The first and second shockwave have merged into a new shockwave that has a speed equal to the slope from traffic state 3 to traffic state 2. The two remaining shockwaves meet each other at time t_5, creating a (forward moving) shockwave, which reaches the end of the link at time t_6.

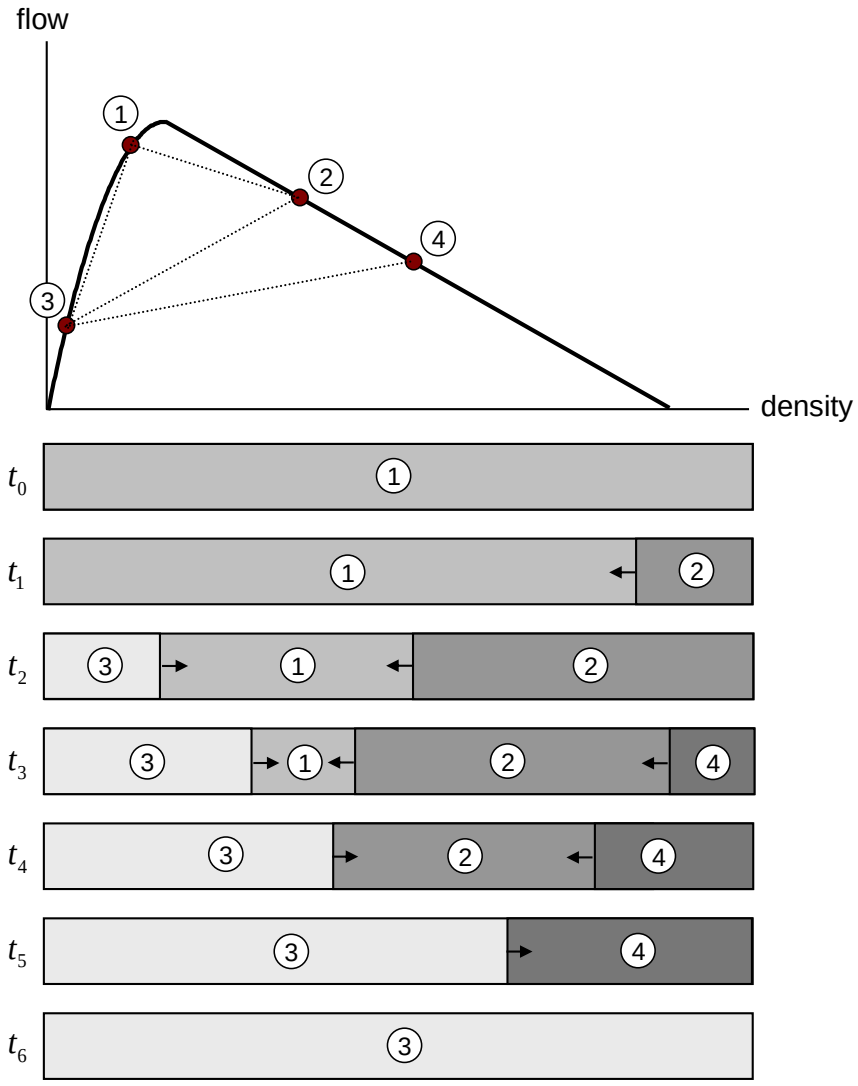

Figure 2: Shockwaves from traffic flow theory

Queuing phase: algorithm

Below the queuing algorithm as currently implemented in a prototype is described. Note that a more recent version of the algorithm is currently being developed and implemented. This newer version is directly derived from the link transmission model (Yperman 2007). Because of new insights, in the new algorithm, it will no longer be necessary to explicitly trace shockwaves through the network, making it less complex and thus faster. Input for the queuing algorithm consists of link inflows ($q_a^{in} \forall a \in A$), link outflows ($q_a^{out} \forall a \in A$) and turn flows ($tf_{ab} \forall a \in A$) as determined in the squeezing phase.

Step 1: Initialize

- Calculate densities using fundamental diagram:

$$k_a^{in} = k_a(q_a^{in} \mid uncongested) \qquad \forall a \in A$$

$$k_a^{out} = \begin{cases} k_a(q_a^{out} \mid uncongested) & \forall a \in A \mid q_a^{in} = q_a^{out} \\ k_a(q_q^{out} \mid congested) & \forall a \in A \mid q_a^{in} > q_a^{out} \end{cases}$$

- Set time, and exodus indicator:
  t:=0
  $exodus = false$
- Create link-event lists SW_a consisting of [t, x, v, k, s] :
  $$SW_a = \begin{bmatrix} 0 & 0 & 0 & k_a^{in} & \infty \\ 0 & l_a & 0 & k_a^{out} & \infty \end{bmatrix} \qquad \forall a \in A$$
- Calculate turn fractions from turn flows
  $tfr_{ab} = tf_{ab} / \sum_{b'} tf_{ab'}$

Step 2: Main loop

- Calculate changes in density and flow and check on stop criterion:
  $\Delta q_a^{last} = q(k_a^{last}) - q(k_a^{last-1})$ $\qquad$ $\Delta q_a^{first} = q(k_a^{first+1}) - q(k_a^{first})$
  $\Delta k_a^{last} = k_a^{last} - k_a^{last-1}$ $\qquad$ $\Delta k_a^{first} = k_a^{first+1} - k_a^{first}$

  If $s_a^j = \infty \wedge \Delta q_a^{last} = 0 \wedge \Delta q_a^{first} = 0 \wedge \Delta k_a^{last} = 0 \wedge \Delta k_a^{first} = 0 \qquad \forall a \in A, \forall j \in J_a$
  Stop! End of algorithm
  End
- Add backward shockwaves to SW_a $\quad \forall a \in A \mid \Delta q_a^{last} * \Delta k_a^{last} \in R^-$

  $$v = \frac{\Delta q_a^{last}}{\Delta k_a^{last}}$$
  $$s = \min\left\{ t + \frac{l_a - (x_a^{last-1} + (t - t_a^{last-1}) * v_a^{last-1}}{v_a^{last-1} - v}, t + \frac{l_a}{-v} \right\}$$
  $$s_a^{last-1} = s \qquad if\ v_a^{last-1} \neq 0 \wedge t^* \neq t + \frac{l_a}{-v}$$
  $$SW_a^{last+1} = \begin{bmatrix} t & l_a & k_a^{last} & v & s \end{bmatrix}$$
- Add forward shockwaves to SW_a $\quad \forall a \in A \mid \Delta q_a^{first} * \Delta k_a^{first} \in R^+$

  $$v = \frac{\Delta q_a^{first}}{\Delta k_a^{first}}$$
  $$s = \min\left\{ t + \frac{x_a^{first+1} + (t - t_a^{first+1}) * v_a^{first+1}}{v - v_a^{first+1}}, t + \frac{l_a}{v} \right\}$$
  $$s_a^{first+1} = s \qquad if\ v_a^{first+1} \neq 0 \wedge s \neq t + \frac{l_a}{v}$$
  $$SW_a^{last+1} = \begin{bmatrix} t & 0 & k_a^{first+1} & v & s \end{bmatrix}$$
- Sort SW_a on x, then on v $\quad \forall a \in A$
- Update shockwaves and time indicator
  $t = \min_{a \in A}\{\min_j\{s_a^j\}\}$ (in case of merging shockwaves t is updated later on)

$$a^* = \arg\min_a \{t\}$$

$$j^* = j \ni (t_{a^*}^{j^*} = t)$$

$$v^* = v^j$$

Backward shockwave meets beginning of link:

$$\left.\begin{aligned} k_{a^*}^{first} &= k_{a^*}^{first+1} \\ SW_{a^*} &= \left\{SW_{a^*} \setminus sw_{a^*}^{first+1}\right\} \\ n^* &= anode_{a^*} \end{aligned}\right\} if\ v^* < 0 \wedge j^* = 2$$

Forward shockwave meetst end of link:

$$\left.\begin{aligned} k_{a^*}^{last} &= k_{a^*}^{last-2} \\ SW_{a^*} &= \left\{SW_{a^*} \setminus sw_a^{last-1}\right\} \\ n^* &= bnode_{a^*} \end{aligned}\right\} if\ v^* > 0 \wedge j^* = last - 1$$

Forward shockwave merges with preceding shockwave:

$$\left.\begin{aligned} \Delta t &= \frac{(x_{a^*}^{j^*+1} - x_{a^*}^{j^*}) + (t_{a^*}^{j^*} - t_{a^*}^{j^*+1}) * v_{a^*}^{j^*+1}}{v_{a^*}^{j^*} - v_{a^*}^{j^*+1}} \\ v &= \frac{q(k_a^{j^*-1}) - q(k_a^{j^*+1})}{k_a^{j^*-1} - k_a^{j^*+1}} \\ k &= k^{j^*+1} \\ t &= t_{a^*}^{j^*} + \Delta t \\ x &= x_{a^*}^{j^*} + v_{a^*}^{j^*} * \Delta t \\ SW_{a^*} &= \left\{SW_{a^*} \setminus sw_a^{j^*+1}\right\} \\ s &= \begin{cases} t + \dfrac{x_a^{j^*+1} - x}{v - v_a^{j^*+1}} & if\ v > 0 \\ t + \dfrac{x - x_a^{j^*-1}}{v - v_a^{j^*-1}} & if\ v < 0 \end{cases} \\ SW_{a^*}^{j^*} &= \begin{bmatrix} t & x & k & v & s \end{bmatrix} \end{aligned}\right\} if\ \ v^* > 0 \wedge j^* \neq last - 1$$

Backward shockwave merges with previous shockwave:

$$
\left.
\begin{aligned}
&\Delta t = \frac{(x_{a^*}^{j^*} - x_{a^*}^{j^*-1}) - (t_{a^*}^{j^*-1} - t_{a^*}^{j^*}) * v_{a^*}^{j^*-1}}{v_{a^*}^{j^*-1} - v_{a^*}^{j^*}} \\
&v = \frac{q(k_a^{j^*-2}) - q(k_a^{j^*})}{k_a^{j^*-2} - k_a^{j^*}} \\
&t = t_{a^*}^{j^*} + \Delta t \\
&x = x_{a^*}^{j^*} + v_{a^*}^{j^*} * \Delta t \\
&k = k_{a^*}^{j^*} \\
&SW_{a^*} = \left\{ SW_{a^*} \setminus sw_a^{j^*} \right\} \\
&s = \begin{cases} t + \dfrac{x_a^{j^*} - x}{v - v_a^{j^*}} & if\ v > 0 \\ t + \dfrac{x - x_a^{j^*-2}}{v_a^{j^*-2} - v} & if\ v < 0 \end{cases} \\
&SW_a^{j^*-1} = \begin{bmatrix} t & x & k & v & s \end{bmatrix}
\end{aligned}
\right\} if\ (v^* < 0 \wedge j \neq 2)
$$

- Check for endtime:

  If $t > endtime \wedge exodus = false$

  $t = endtime$

  $k_a^{first} = 0 \qquad \forall a \in A$

  $exodus = true$

  Break: start with new loop

  end

- Perform node model on node n* (if n* is determined in 'update shockwaves' and n* is not a centroid)

$$
G_{ab} = \min_{b'} \left\{ \frac{q(k_{b'}^{first})}{\sum_{a'} q(k_{a'}^{last}) * tf_{a'b'}} * q(k_a^{last}) tfr_{ab} \ ,\ q(k_a^{last}) tfr_{ab} \right\} \forall ab \mid bnode_a = n^* \wedge anode_b = n^*
$$

$$
\left.
\begin{aligned}
&\chi = \begin{cases} 1 & if\ k_a^{last} > K_a \\ 0 & if\ k_a^{last} \leq K_a \end{cases} \\
&q_a^{out} = \sum_{b \in A \mid anode_b = n^*} G_{ab} \\
&k_a^{last} = k_a(q_a^{out} \mid \chi)
\end{aligned}
\right\} \qquad \forall a \in A \mid bnode_a = n^*
$$

$$
\left.
\begin{aligned}
&\chi = \begin{cases} 1 & if\ k_a^{first} > K_a \\ 0 & if\ k_a^{first} \leq K_a \end{cases} \\
&q_a^{in} = \sum_{a' \in A \mid bnode_{a'} = n^*} G_{a'a} \\
&k_a^{first} = k_a(q_a^{in} \mid \chi)
\end{aligned}
\right\} \forall a \in A \mid anode_a = n^*
$$

- Return to step 2

3.4 STAQ: derivation of travel times

Whenever the inflow rate or outflow rate on a link changes (due to an event at the beginning or end of a link), the algorithm of the queuing phase stores the cumulative link inflow and cumulative link outflow of this link. These cumulative inflow and outflows can be used to derive travel times on any given point in the time studied. Because the queuing phase continues until all traffic demand has reached its destination, travel times can also be calculated for traffic that reaches its destination after the study period has come to an end.

The way STAQ calculates travel times is very similar to the way dynamic traffic assignment models calculate travel times based on cumulative flows as described in Newell (1993). However, because STAQ lacks a true time variable, like all static assignment models, traffic travels through its path instantaneous. This means that in the model vehicles are on every link on their path at the same time. For the calculations of shockwaves (thus delays) however, there does exist a time instant variable. When drawing cumulative flow curves, this lack of a time variable when travelling free flow through the network means that differences between cumulative inflow and cumulative outflow curves in STAQ represent delays, not travel times like in true DTA models. This means that in order to calculate link travel times, the free flow travel time should be added to the delay derived from the cumulative flow curves.

4. APPLICATION OF STAQ PROTOTYPE

A fully working prototype of the STAQ algorithm has been developed. In this section results of the prototype on different networks are presented. First, runs on two test networks are presented which show how the algorithm works and that spillback and queuing is taken into account. Then, results of runs on the network of Amsterdam are presented in order to give insight in the real life performance of the STAQ algorithm.

4.1 Corridor network

The corridor network has already been introduced in section 2.5; figure 1. The corridor network is the simplest network tested. There are no diverging or splitting points, only bottlenecks. Network capacities are quantified as 1000 veh/h per lane and traffic demand is quantified as 2200 veh/h. Since this network contains no route choice options assigning all or nothing yields the Beckmann solution as shown in figure 3. This is the quantification of figure 1.

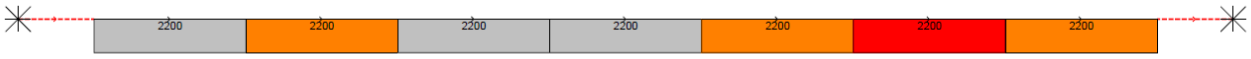

Figure 3: Corridor network with quantified flows using Beckmann assignment

After applying the squeezing algorithm to this network the inflows and vertical queues are calculated as displayed in figure 4. The outcomes are as expected: the flow of 2200 is lowered whenever a link with insufficient capacity is encountered on the path, forming vertical queues the size of the difference between flow and capacity at the beginning of the bottleneck link.

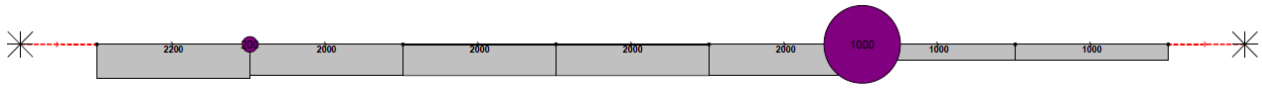

Figure 4: Inflows (bars) and vertical queues (pies) on corridor network after squeezing phase

Applying the queuing algorithm to the corridor network using inflows as displayed in figure 4 yields an initial situation (figure 9), six events (figures 10 to 15) and a final situation (figure 16). During the first five events, queues grow on the first and fifth link of which the latter causes spillback onto upstream links. At the sixth event both queues merge and in the final situation inflow of all links upstream from the bottleneck in link six is restricted to this bottleneck. This run demonstrates how STAQ handles the different events sequentially producing inflows and travel times at all events. The growing and merging of backward queues was also demonstrated.

4.2 Network with crossing paths

The network with crossing paths is used to demonstrate interaction between different paths. Like in the corridor network, there are no route choice options, just bottlenecks. This network however, has two paths which cross each other at an intersection, making it possible to demonstrate the behaviour of the node model with independent streams. The network and its link capacities are shown on the left hand of figure 6. Free flow speeds are set to 80 km/h on all links. Traffic demand on the path from centroid 11 to centroid 12 is 4000 whilst traffic demand on the path from centroid 13 to centroid 14 is 2000. Assigning all or nothing yields the Beckmann solution as shown in the right hand of figure 5.

After applying the squeezing algorithm to this network the inflows and vertical queues are calculated as displayed in figure 6. Two vertical queues appear: one on the path of centroid 11 to centroid 12 (500 in vertical queue) and one on the path of centroid 13 to centroid 14 (3000 in vertical queue).
Applying the queuing algorithm to the network with crossing paths using inflows as displayed in figure 6 yields an initial situation, five events and a final situation. These are displayed in figures 17 to 20. It can be seen that the bottleneck on the path from centroid 11 to centroid 12 starts blocking the intersection at the second event, in which the path from centroid 13 to centroid 14 is also blocked. This creates a forward shockwave from the intersection towards centroid 14 which starts to dissolve the queue at the third event. This queue has completely disappeared at the fifth event. This run shows how STAQ handles interaction between different paths over intersections, forming both backward and forward shockwaves.

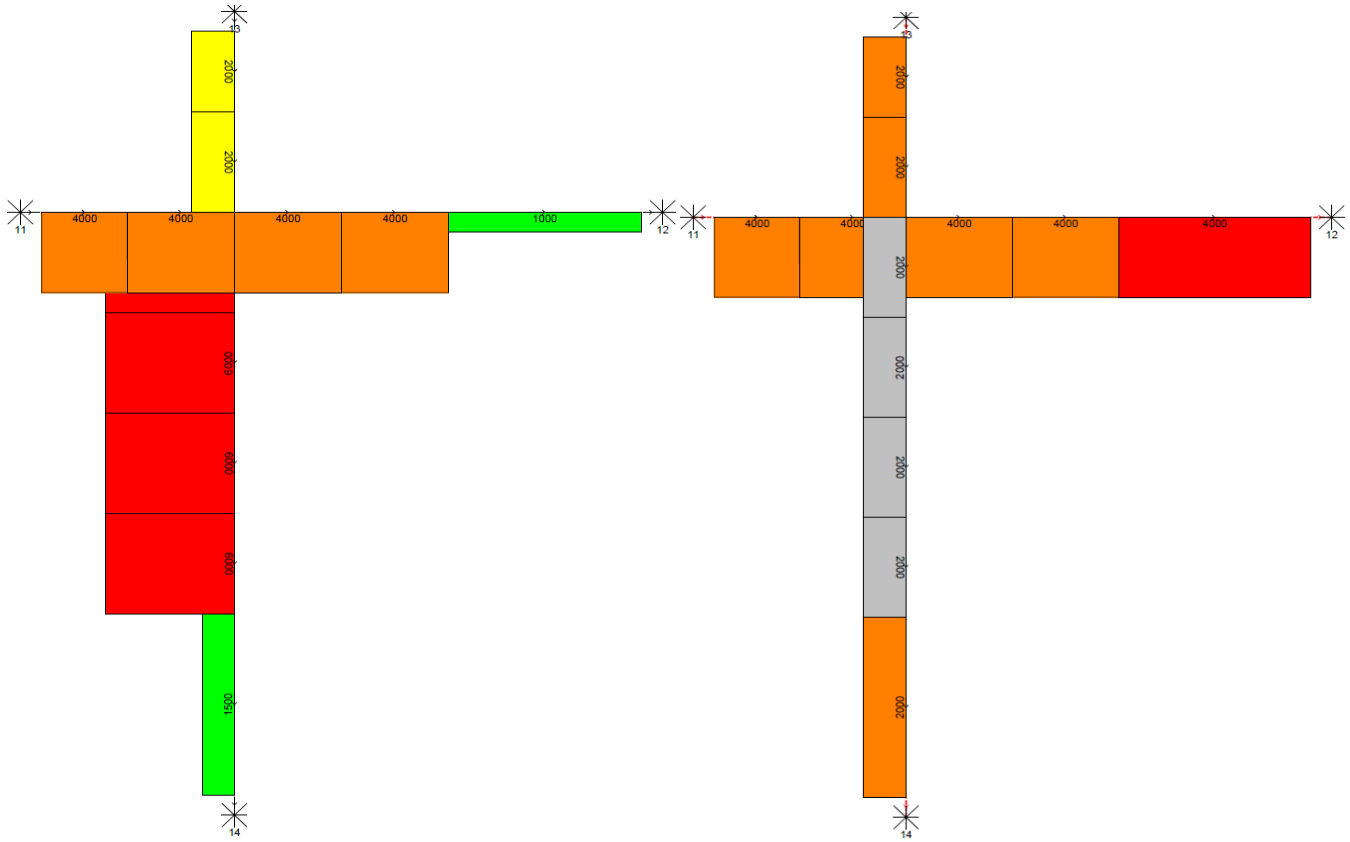

Figure 5: Capacities (left) and Beckmann flows (right) on network with crossing paths

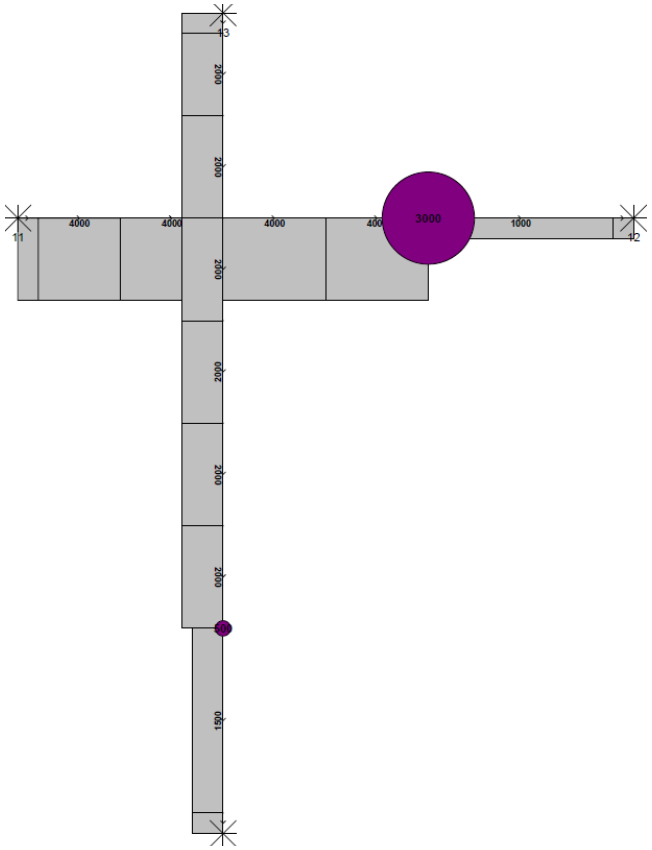

Figure 6: Inflows (bars) and vertical queues (pies) on network with crossing paths after squeezing phase

4.3 Amsterdam network

The algorithm has also been applied on the network of Amsterdam consisting of 3868 links and 279 centroids in order to give insight in the scalability and calculation speed of the algorithm. Table 1 displays some properties of the Amsterdam network.

Indicator	Value
# links	3868
# centroids	279
# Hbpairs	77562
# used paths	72908
# blocked paths	24056
average #links per per path	45.65
# squeezing increments	49
# of queues after squeezing	23
average #connectinglinks per node	1.27
average absolute shockwave speed	$5^{(2)}$
duration of queuing phase	1
average link length	0.1491

Table 1: properties of the Amsterdam network

The squeezing phase needed 49 increments to load all traffic onto the network resulting in inflows and queues as displayed in figure 7. Application of the queuing phase on the Amsterdam network yields 23 initial shockwaves caused by the vertical queues displayed in figure 7 triggering 2216 events. Although the number of events seems very large, calculations that need to be done per event are very limited because each event only causes calculations with respect to one node and the links connecting to that node while the rest of the network is left untouched.

Figure 8 shows results after 1 hour of STAQ (evening peak). The results have not been analysed on plausibility of the calculated queues and travel times. This was not done since route choice was based on an all or nothing assignment because multiple paths per OD pair are not yet supported by the prototype.

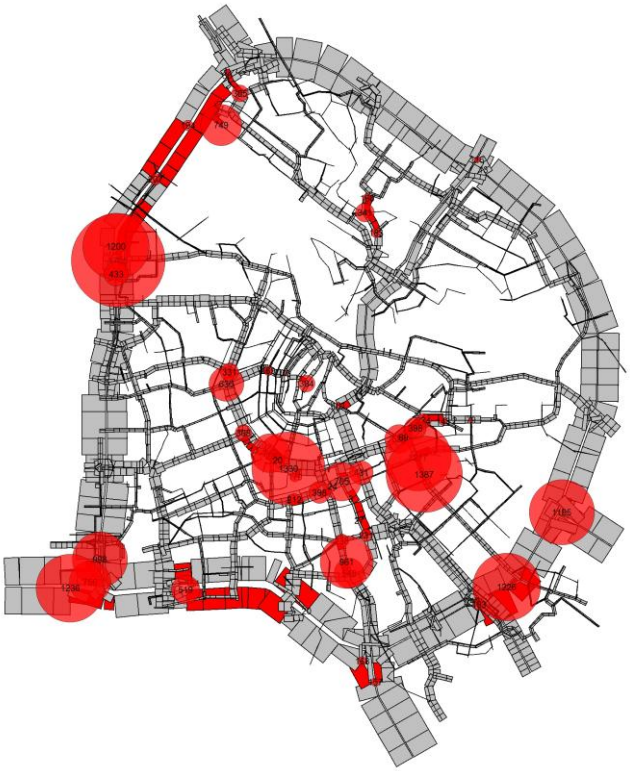

Figure 7: Inflows (bars) and vertical queues (pies) on the Amsterdam network after squeezing phase

Figure 8: inflows (width) and speed relative to free flow speed (colour) on the Amsterdam network after 9 minutes of queuing

In order to give insight in the computational complexity, below the amount of computational time needed to solve each of the two problems in STAQ is related to an all or nothing assignment.

In an all or nothing assignment, the amount of calculation time required is proportional to the number of paths times the average number of links per path (equation 15).

$$T_{allornothing} \propto P * \underset{p \in P}{avg}(A_p) \tag{15}$$

The calculation time needed for the STAQ squeezing phase is proportional to the number of increments needed, the number of paths and the average number of links per path (16). The latter is divided by two because links behind blocked paths are removed from the path which means, assuming a uniform spread of blocked links over the network, that the average number of links per path to be processed will be half of the average number of links per path. The number of increments *I* is dependent on the crowdedness of the network. The more blocking links (links on which demand exceeds capacity), the more increments needed. This effect is tempered when multiple blocking links exist on the same path because existence of blocking links upstream result in lower demand on links downstream, thereby possibly removing such a downstream blocking link.

$$T_{squeezing} \propto I * \overline{P} * \underset{p \in P}{avg}(A_p / 2) \tag{16}$$

The calculation time needed for the STAQ queuing phase is proportional to the number of events created. The number of events depends on the number of vertical queues created during the squeezing phase, the average number of incoming and outgoing links per node, the length of the chosen study period, the average absolute shockwave speed and the average link length. Proportionality is then calculated using (17).
The average number of connecting links per node is network dependent and determines how many new shockwaves are added on average when an existing shockwave reaches a node[1]. The average link length divided by the absolute shockwave speed determines how much time it takes for a shockwave to reach the end of a link, thus creating new shockwaves. The length of the chosen study period is equal to the parameter endtime, assuming that simulation starts at time=0. In (17) the parameter endtime represents the amount of time that the queuing phase has to create events.

$$T_{queuing} \propto E \propto queues * \left(\frac{endtime/2}{\underset{a \in A}{avg}(l_a) / \underset{a \in A, j \in J_a}{avg}(\left|\omega_a^j\right|)} \right)^{\underset{nodes}{avg\,(connectinglinks)}} \tag{17}$$

Table 2 shows the number of calculations needed for an all or nothing assignment and the two phases of STAQ on the Amsterdam network. The

figures in table 2 are based on (15), (16) and (17) and the properties of the Amsterdam network as displayed in table 1.

It can be seen that the number of calculations needed for the squeezing phase of STAQ is roughly 8 times bigger than an all or nothing assignment[3]. The STAQ queuing phase needs only 0.025% of the number of calculations needed in an all or nothing assignment. It must be noted that the number of calculations needed for the STAQ queuing phase is not directly comparable to the number of calculations needed for the other assignments because the STAQ queuing phase performs calculations on more links (the current link and its connecting links) and it performs a node model.

	All or nothing	STAQ squeezing	STAQ queuing
Estimated number of calculations	3.328.250	26.904.832	826
Unit	change of inflow on 1 link	change of inflow on 1 link	change of density on 1+connectinglinks, perform node model
Index	100	808	0.025

Table 2: number of calculations on the Amsterdam network for an all or nothing assignment and the two phases of STAQ

5. DISCUSSION AND FURTHER WORK

Based upon the limited experience with STAQ, in terms of both realism of outcomes and calculation times, STAQ should be positioned in between static and dynamic assignment. Because the current implementation STAQ is still a prototype no definite judgement can be made about either of these criteria. The algorithms will be further optimized and developed and only application in real life transport studies can prove if STAQ outcomes meet demanded levels of realism. It is clear however, that STAQ is methodological superior to any of the existing static and semi dynamic models presented in chapter 2.

Below recommendations for further research related to the development of STAQ are presented.

Meaning of STAQ travel times on a given time instant
As mentioned earlier, lack of a time variable in the squeezing algorithm has implications on the study period when applying the queuing algorithm. Because the queuing algorithm starts with vertical queues (with a queue length of 0) taken from the squeezing algorithm it implicitly assumes that there are no queues present at t0. This means that the study period over which STAQ is applied must be chosen in such a way that there are (more or less) no queues in reality at t0. For the end of the study period the lack of a time variable means that STAQ implicitly assumes that there are no more growing queues in reality at the end of the study period. As a result of this, travel times on any given time instant should not be used for analysis. Travel times which can be clearly interpreted are:

- the average travel time calculated using cumulative flows based on the total results of the queuing phase. These travel times represent the

average travel time any user will experience when travelling during the study period.
- the travel time at the end of the study period, which represents the maximum travel time users can encounter when travelling during the study period.

Node model
The node model that is now applied in STAQ is based on Bliemer (2007). Tampère et all (2010) state that this node model violates the invariance principle as defined by Lebacque and Khoshyaran (2005). The invariance principle was introduced to avoid discontinuous changes in the flows. The principle states that under constant demand and supply constraints, flows should be invariant during an infinitesimal time step. The effects of this violation of the invariance principle on STAQ will be investigated further.

Junction modelling
So far, only delays caused by capacity constraints on links are incorporated into STAQ. Delays caused by intersections (nodes) are not due to the lack of junction modelling. When STAQ is to be applied in urban transport models, junction modelling should be added. How this should be done, and especially the interaction between the junction modelling and STAQ will be investigated further.

Application and optimization within route choice loop
The test runs presented in this paper all used route choice based on an earlier performed all-or-nothing assignment. Application of STAQ within a route choice loop in order to reach Wardrop's user equilibrium has yet to be done. Different approaches can be considered varying from fully incorporating STAQ within the route choice loop to applying a single run of STAQ after a static equilibrium assignment.

NOTES

[1] More precisely, if one would want to calculate the average number of connecting links per node as displayed in equation (17), the averaging should only take place over nodes that will be encountered by one or more shockwaves during the queuing phase. Since this is not known beforehand this cannot be done.

[2] Average absolute shockwave speed has been estimated, not calculated

BIBLIOGRAPHY

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

Beckmann, M., McGuire, C. & Winsten, C. (1956). *Studies in the economics of transportation*. Yale University Press, New Haven, CT.

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

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

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* **37** () 501-519.

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

Chen, H.-K. and C.-F. Hsueh (1998) A Model and an Algorithm for the Dynamic User-Optimal Route Choice Problem. *Transportation Research B*, **32** (3) 219–234.

Daganzo, C.F. (1977a) On the traffic assignment problem with flow dependent costs--I, *Transportation Research*, **11** (6) 433-437.

Daganzo, C.F. (1977b) On the traffic assignment problem with flow dependent costs--II, *Transportation Research*, **11** (6) 439-441.

Daganzo, C.F. (1994) The Cell Transmission Model: A Dynamic Representation of Highway Traffic Consistent with the Hydrodynamic Theory. *Transportation Research B*, **28** (4) 269–287.

Janson, B.N. (1991) Dynamic Traffic Assignment for Urban Road Networks. *Transportation Research B*, **25** (2/3) 143–161.

Larsson, T. & Patriksson, M. (1995). An augmented lagrangean dual algorithm for link capacity constrained traffic assignment problems. *Transportation Research B*, **29** () 433-455.

Lebacque, J.P., Khoshyaran, M.M., 2005. First-order macroscopic traffic flow models: intersection modeling, network modeling. *In: Proceedings of the 16th International Symposium on Transportation and Traffic Theory (ISTTT), pp. 365–386.*

Marcotte, P, Nguyen, S and Schoeb, A. (2003). A strategic flow model of traffic assignment in static capacitated networks, *Operations Research* **52** () 191-212

Nagurney, A. (1993) *Network economics: a variational inequality approach.* Kluwer Academic Publishers, Boston, USA.

Nesterov, Y. (2000). Stable traffic equilibria: properties and applications. *Optimization and Engineering*, **3** () 29-50.

Nesterov, Y. & de Palma, A. (2000a). Stable dynamics in transportation systems. *CORE discussion paper*.

Nesterov, Y. & de Palma, A. (2000b). Stable dynamics transportation systems. *In: Proceedings of the European Transport Conference 2000*.

Nesterov, Y. & de Palma, A. (2003). Stationary dynamic solutions in congested transportation networks: summary and perspectives. *Network and Spatial Economics*, **3** () 371-395.

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 B* **27** () *281-313*.

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

Prashker, J. & Toledo, T. (2004). A gradient projection algorithm for side-constrained traffic assignment. *European Journal of Transport and Infrastructure Research*, **4** (2) 177-193.

Ran, B., H.K. Lo and D.E. Boyce (1996) A Formulation and Solution Algorithm for a Multi-Class Dynamic Traffic Assignment Model. In: J.-B. Lesort (ed.), *Transportation and Traffic Theory*, Proceedings of the 13*th* International Symposium on Transportation and Traffic Theory in Lyon, Pergamon, Elsevier, pp. 195–216.

De Romph, E. (1994) *A dynamic traffic assignment model: Theory and applications.* PhD Thesis, Delft University of Technology, Delft, The Netherlands.

Shahpar, A., Aashtiani, H. & Babazadeh, A. (2008). Dynamic penalty function method for the side constrained traffic assignment problem. *Applied Mathematics and Computation*, **206** () 332-345.

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

Smulders, S. (1988) Modelling and Filtering of Freeway Traffic Flow. *Report OSR8706*, Centre of Mathematics and Computer Science, The Netherlands.

Tampère, C.M.J., et al. (2010) A generic class of first order node models for dynamic macroscopic simulation of traffic flows. *Transportation Research B (2010), doi:10.1016/j.trb.2010.06.004*

US Bureau of Public Roads (1964). *Traffic assignment manual*. US Bureau of Public Roads, US Government Printing Office, Washington DC.

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

Yang, H., Yagar, S. (1994). Traffic assignment and traffic control in general freeway-arterial corridor systems, *Transportation Research B*, **28** (6) 463-486.

Yang, H., Yagar, S. (1995), Traffic assignment and signal control in saturated road networks, *Transportation Research Part A*, **29** (2) 125-139.

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

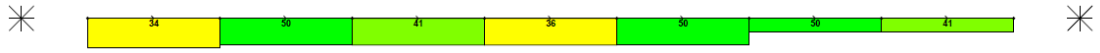

Figure 9: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t=0 (initial situation)

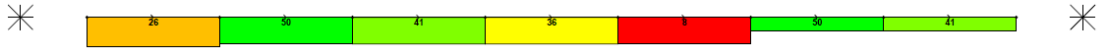

Figure 10: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t=0.08215 (first event)

Figure 11: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t=0.24259 (second event

Figure 12: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t = 0.42677 (third event)

Figure 13: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t = 0.50286 (fourth event)

Figure 14: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t = 0.58500 (fifth event)

Figure 15: Link speeds (colour, figures) and inflows (bandwiths) on corridor network at t = 0.65000 (sixth event)

Figure 16: Link speeds (colour, figures) and inflows (bandwiths) on corridor network after t= 0.65000 (until t=1.0)

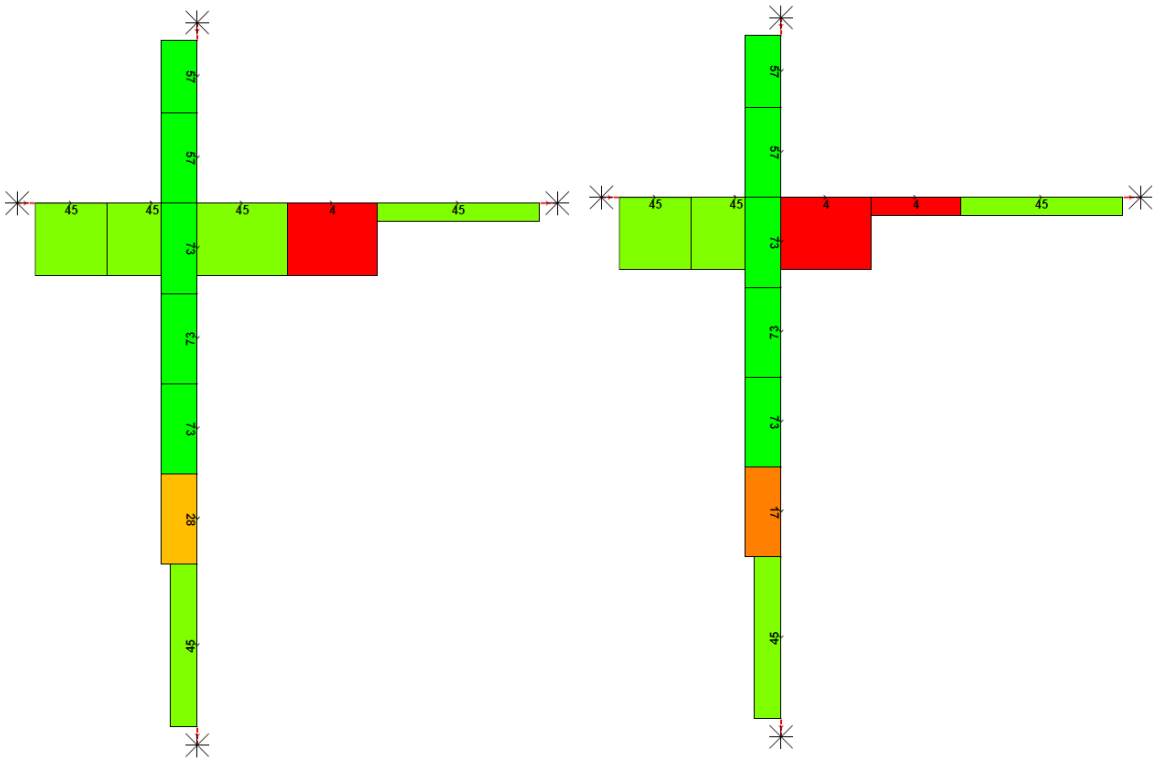

figure 17: link speeds and inflows on network with crossing paths: initial situation and first event

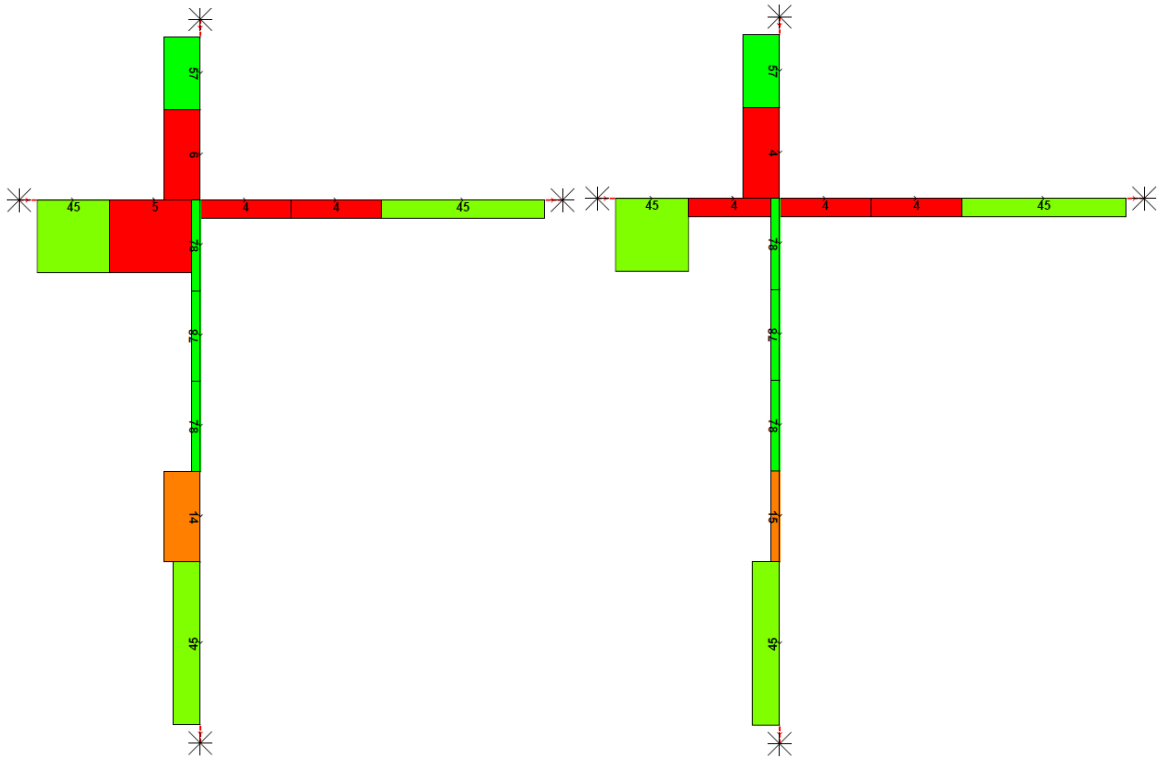

figure 18: link speeds and inflows on network with crossing paths: second and third event

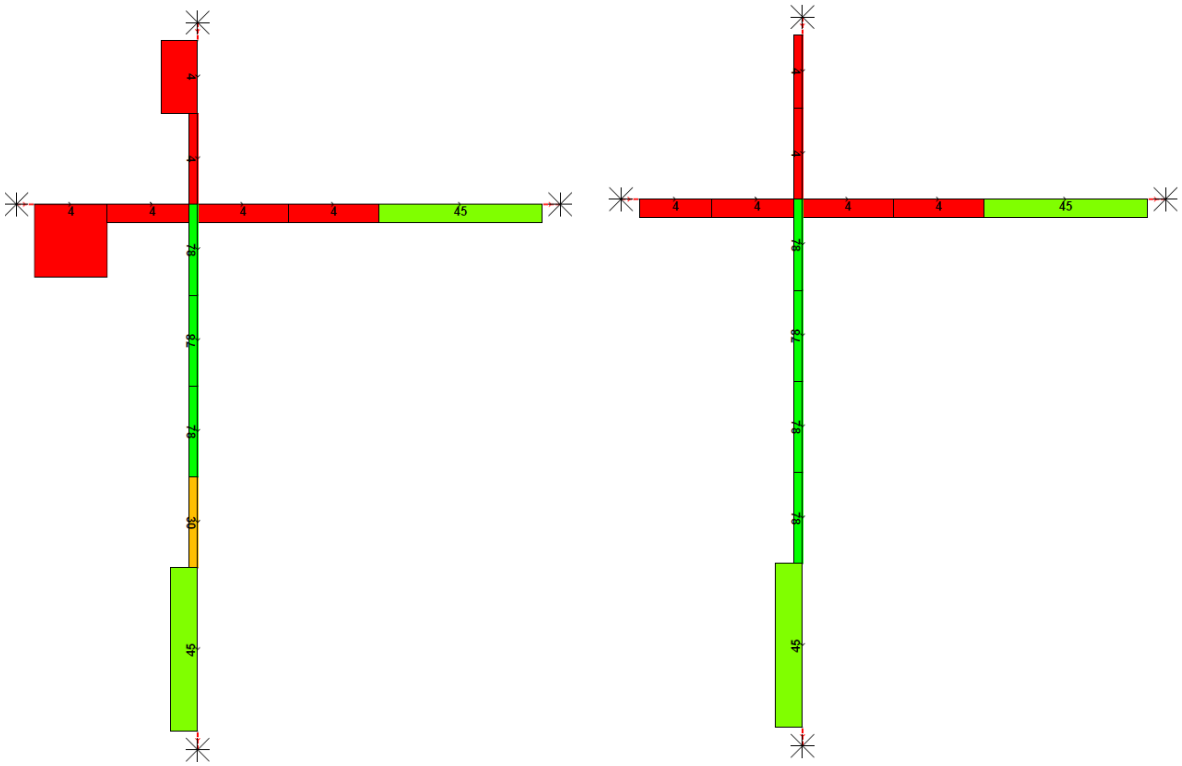

figure 19: link speeds and inflows on network with crossing paths: fourth and fifth event

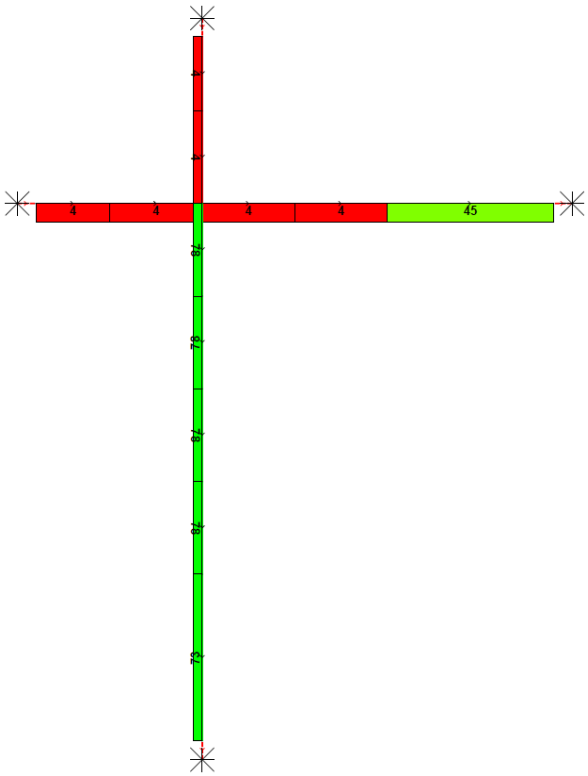

figure 20: link speeds and inflows on network with crossing paths: final situation

Appendix: Fundamental diagram and traffic flow theory

In STAQ the Smulders' fundamental diagram (see Smulders, 1988, or De Romph, 1994) is used, which is given by the following equation:

$$q_a(k_a) = \begin{cases} k_a\left(V_a - \left(V_a - \dfrac{C_a}{K_a}\right)\dfrac{k_a}{K_a}\right), & \text{if } 0 \le k_a \le K_a; \\ \dfrac{C_a(J_a - k_a)}{J_a - K_a}, & \text{if } K_a \le k_a \le J_a, \end{cases} \qquad \text{(x14)}$$

where k_a is the density (km/h) on link a, and V_a, K_a, and J_a the given maximum speed, critical density (which is the density at capacity C_a), and the jam density of link a, respectively.

It consists of a free-flowing branch and a congested branch. The simplified fundamental diagram as proposed by Newell (1993) and applied in the link transmission model by Yperman (2007), assumed both branches to be linear (indicated by the dashed blue line in Figure 2). While this linear assumption for the congested branch is often assumed to be close enough to reality, a linear line for the free-flowing part is somewhat unrealistic. For example, if the free-flow speed is 120 km/h, then anything below this speed is assumed to be in congestion. Therefore, we assume the free-flowing branch to be quadratic, following Smulders' fundamental diagram. Should this be needed, Newell's diagram can still easily be implemented by setting $K_a = C_a / V_a$.

The colours in Figure 1 refer to the link speeds as indicated in Figure 21, which is 'green' in free-flow conditions, 'yellow' at capacity, and different shades of 'orange' to 'red' for more congested situations.

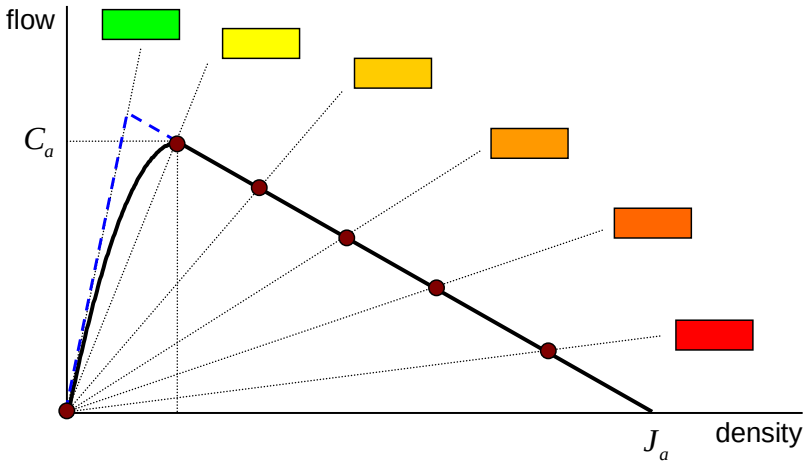

Figure 21: Smulders' fundamental diagram used in STAQ and speed colours

The fundamental diagrams are assumed to be given for each link in the network, and are used in conjunction with regular traffic flow theory to derive queues and shockwaves. Figure 22 illustrates an example of a link upstream of a bottleneck with a given link inflow and outflow. This inflow and outflow correspond to two points on the fundamental diagram, as indicated in Figure 22. Due to the bottleneck the outflow is restricted to the inflow capacity into the bottleneck. Since the inflow is higher than the outflow, a queue will build up. The queuing speed is indicated by the slope from the line from the origin to the second point on the fundamental diagram. The first part of the link will initially remain free-flow with a speed equal to the slope of the line from the origin to the first point on the fundamental diagram. However, as time elapses, the queue builds up and may reach the beginning of the link, such that over time the traffic conditions at the beginning of the link switch from the traffic situation in point 1 to the traffic situation in point 2. The time it takes the queue to spillback to upstream links (provided that the inflows and outflows are stationary, as assumed in a static model) is determined by the shockwave speed, which is the slope of the line from point 2 to point 1. Given the link length, the time that spillback occurs can readily be computed.

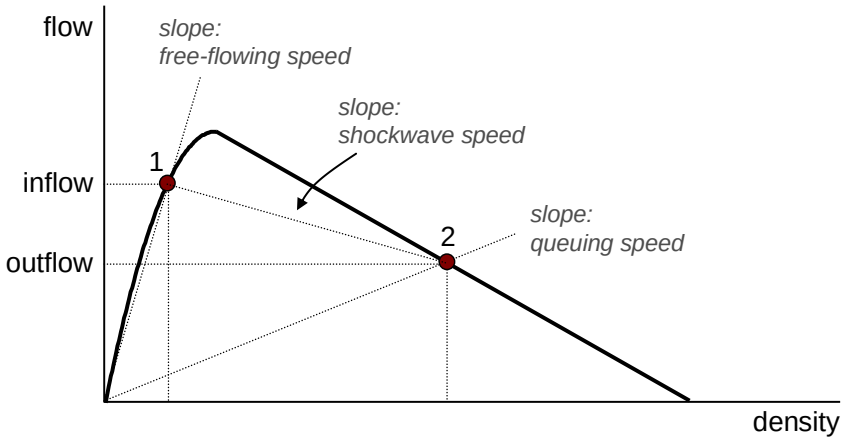

Figure 22: Shockwaves from traffic flow theory