MODELLING QUEUES IN STATIC TRAFFIC ASSIGNMENT

Michael Bundschuh
Peter Vortisch
PTV AG, Karlsruhe, Germany
Tom Van Vuuren
Mott McDonald, UK

1 INTRODUCTION

In the past several approaches were made to model blocking back and downstream metering effects of bottlenecks in assignment procedures while preserving the computational advantage of static assignment. One classic is the method implemented in SATURN where a static assignment is complemented by a traffic flow simulation in parts of the network [Van Vliet 1982]. Another approach was proposed by Bell as an extension of the path flow estimator method [Bell 1996]. The so-called quasi-dynamic PFE computes a static assignment for a series of time slices and hands over queues from one time slice to the next. The result is an improved modelling of queues, but not yet a realistic model of the propagation of queues in the network or the metering effects of bottlenecks.

The objective of the method proposed in this paper is to include congestion phenomena in the context of a static assignment and thus to achieve a more realistic modelling of congested networks but to avoid the computation time penalty induced by fully dynamic assignment methods, especially those incorporating flow simulation. The proposed method is not an assignment procedure in the pure sense since it does not compute routes and volumes. Instead, the result of a static assignment is modified. In the following text it is called pseudo-dynamic assignment. Of course it is necessary to assume a time period for the static assignment.

2 PSEUDO-DYNAMIC ASSIGNMENT

2.1 Basic idea

The pseudo-dynamic assignment is supposed to bridge the gap between static and dynamic assignment procedures. The aim is a procedure which requires significantly less computing effort than a dynamic assignment but which is on the other hand capable to reflect spill back phenomena and their impact on capacity.

The idea of the proposed method is simple: travel demand is redistributed along routes computed by any static assignment procedure beforehand. In a first phase fractions of the volume of each route are passed from one link to
the next along the route until capacity of a link is reached. This propagation of volumes obeys the following rules:

1. The traffic volume on each link is limited by the capacity of the link. The model assumes the bottleneck at the end of the link, i.e. the capacity restricts the outflowing volume.

2. The queue length on each link is limited by the queuing capacity of the link.

3. Traffic can not flow across a congested link (i.e. a link with a queue), even if the traffic follows a route that does not include the bottleneck causing the congestion.

4. The inflow of a link is limited by the sum of capacity and queuing capacity.

Rule 3 is relaxed by the introduction of a permeability factor $P$ (0..1). The permeability describes which fraction of the volume of a route not including the bottleneck is allowed to pass the congested link. The factor is included in order to model the influence of separate turning lanes. For $P = 0$ rule 3 is enforced strictly.

The second phase of the method computes the delay times caused by congested links. During this phase no further traffic is fed to the network. Only the traffic stored in the queues is propagated along the routes according to the same rules as before. This is done in $N$ small time slices. The second phase ends when all queues are dissolved and no more traffic is in the network. The result is a series of $N$ snapshots of the congestion situation from which the delay times can be computed.

2.2 Description of the method

Phase 1: flow propagation and queue building

In the first phase of the procedure traffic flows into the road network. It follows the routes computed by a previously computed static assignment, but behaves according to the rules 1 to 3 listed above. It is assumed that the assignment is valid for a time period $T$.

To compute this, each route is considered $N$ times and every time the $N$th part of the route volume is propagated from origin to destination. This means that $N$ is the discretization parameter of the method. Traffic can flow freely from one link to the next until a link is reached on which a queue has already formed or whose flow capacity is exceeded. The part of the volume that can not flow out of the link is added to the queue. To do this in small portions instead for the whole volume at once is a common way to avoid dependency of the link processing order. It ensures that the rest capacity of a partially congested link is shared among all preceding (upstream) links.
More formally, the method is described as follows:

For each iteration $i$ from 1 to $N$:

For each route $R$:

- Compute the current part of the route volume as $\text{CurPathVol} = \text{Vol}(R) / N$, where $\text{Vol}(R)$ is the volume on route $R$ computed by the preceding assignment.

- Walk along the links of the route from origin connector to destination connector. For each link:

  1. If the current link is the origin connector:

     The connector is loaded with the route volume. The connector does not have a limiting capacity. Existing queues are not considered since it is assumed that the connector is the abstract representation of several possibilities to enter the network. The full route volume is propagated to the next downstream link.

  2. The current link is not a connector:

     If the link is already congested, i.e. there is a queue ($Q_s > 0$) on the link, the current link volume $\text{Vol}_l$ is the inflowing route volume reduced by multiplying by $P$. (Remember, $P$ is the permeability factor describing which portion of the inflow can pass an already existing queue. For $P = 0$ the link is blocked totally by a queue.) The rest $(1-P)$ of the inflow goes into the queue of the link. The number of additional vehicles in the queue is $(1-P)$ inflow times assignment period $T$. The increase in queue length is then computed as number of vehicles times the average vehicle length in the queue. The average vehicle length in queues is a global procedure parameter that is scaled by the number of lanes of the current link.

     If there is no queue on the current link, the current route volume remains unchanged.

     Then the volume propagated to the next link downstream is determined. The volume is limited by the unused flow capacity $\text{cap}_l$ of the current link and the current route volume.

     $$\text{Vol}_l (\text{new}) := \text{Vol}_l (\text{old}) + \min \{ \text{CurPathVol} (\text{old}), \text{cap}_l - \text{Vol}_l (\text{old}) \}$$

     $$\text{CurPathVol} (\text{new}) := \text{Vol}_l (\text{new})$$

     If there is a remaining difference between inflow $\text{CurPathVol} (\text{old})$ and outflow $\text{CurPathVol} (\text{new})$, it is added to the queue of the current link.
If the queue length $Q_l$ exceeds the queuing capacity $K_l$ of the current link, the difference ($Q_l - K_l$) is shifted to the upstream links along the route. Each link can accept queued vehicles up to its own queuing capacity. The backpropagation of the queue ends at the origin connector at the latest, where unlimited queuing capacity is assumed. During this process, the volume shifted back and placed in a queue of a link is subtracted from the outflow of this link, thus reconstructing the situation as if this volume never had been propagated downstream.

If the outflow of the current link $CurPathVol$ is not zero, the next link downstream becomes the current link. If $CurPathVol = 0$, then the next route is processed.

3. If the current link is the destination connector:

The propagated volume $CurPathVolume$ is added to the volume of the connector and leaves the network. Processing of the current route is finished.

Process next link of route $R$.

Process next route $R$

Process next iteration $i$.

**Phase 2: Queue discharge and delay time distribution**

In order to compute not only queue lengths and corrected volumes but as well lost times, the propagation procedure described above is continued without further inflow into the network until all queues are dissipated completely. All traffic flows in phase 2 are fed from the queues build up in phase 1.

Phase 2 again is processed in iterations. One iteration models the outflow taking place in the $M$th part of the assignment period of phase 1. Therefore the available link flow capacities are reduct to the $M$th part in each iteration; the queuing capacities remain unchanged. His means, that $M$ is the discretization parameter of phase 2. After each iteration $i$ the queue lengths for all links $l$ are recorded in $Q_l(i)$. The value in $Q_l(0)$ represents the queue length on link $l$ at the end of phase 1.

The only temporal dimension available in the context of a static assignment is the assignment period $T$.

The first phase can be considered as a measurement period of length $T$, and each iteration of phase 2 as a measurement period of length $T/M$. At the beginning of phase 1 all queue lengths are zero. The total delay time $D_l$ of a link $l$ is defined as the sum of the delay time values in all measurement periods:
\[ D_i = T^* \left( \frac{D_i(0)}{2} + \frac{\sum D_i(i) + D_i(i-1)}{2M} \right) \]

\( D_i(0) \) is the queue length at the end of phase 1.

The average delay time per vehicle \( t_D \) can be computed by division by the volume of the link. For unloaded links the average delay time is naturally 0.

### 2.3 Integration in Assignment Procedures

The blocking back method described above can be integrated in the context of assignment procedures by considering the computed delay times in the actual travel times on the links. For links without queues the travel time computed by the capacity restraint function from the link volume remains unchanged and for those with queues the travel time is increased by \( t_D \):

\[ t_{cr} = CR(Vol) + t_D. \]

Both terms are necessary since \( t_D \) describes only the delay time in the queue and not the travel time on the (possibly quite long) free section upstream the queue.

If blocking back is computed after a normal static assignment procedure, information about queue lengths and delay times are produced, but there is no feedback to the route choice during the assignment. As long as the road network does not contain alternative routes, the result of the computation will be already quite realistic. But in the most networks, feedback to the route choice model is desirable to get a more realistic assignment result.

In many assignment procedures the total demand is assigned to the links in successive portions. For these assignment procedures the blocking back can be easily integrated by computing it after each iterative step. The computed delay times will then influence the route choice in the next iterative step. Examples of these methods are the incremental assignment or the learning method by Lohse.

For equilibrium methods the integration is more difficult. Since the traditional algorithms are iterative as well, the first approach might be to compute blocking back after each iteration as in the methods above. But this will not work for the following reasons:

- A necessary condition for the convergence of the procedures is that travel time on a link is a monotonous function of the link volume. But the travel times computed by blocking back depending on the situation on other links as well.

- In the context of the equilibrium algorithms the volume is assumed to be the same along the route, but blocking back computation changes the link volumes along the route.
Since equilibrium methods are important in practical application, an integration of the blocking back computation is desirable nevertheless. As an approximation, the following approach can be adopted: After each equilibrium step delay times are computed but instead adding them to the travel times, the capacity restrained function is modified to reflect the higher travel time. The modification must fulfil the following conditions:

- The modifies CR-function depends monotonously on the link volume.
- For volumes close to the volume computed in the last equilibrium step the modified CR function delivers travel times close to the times including the delay times from the blocking back.

Since equilibrium assignment and blocking back use each others results, an additional iteration is induced in the whole procedure.

3 ADDITIONAL REMARKS

3.1 Strengths and weaknesses

The presented approach allows to assess time delays caused by propagating queues in the network in a static assignment as well as the improved situation downstream of bottlenecks without the necessity to use a full dynamic assignment method or a microsimulation method.

The result obviously depends on the processing order of the routes. The influence of the order decreases with the number of incremental steps $N$ of the first phase of the procedure and will be arbitrarily small for large $N$.

In practical applications, normally a small number of increments will be chosen because of computing time reasons. It is important to note that the traffic volume stored in the queues is distributed in the second phase to routes proportionally to the route volumes traversing the link and not according to their original destinations. This means that the movement in the second phase is not the consistent continuation of the movement in the first phase, but the delay time is influenced by all routes in a fair way.

3.2 Computing time issues

The computational effort of one incremental step in both phases is comparable to the update step of the link volumes as it is part of every assignment procedure. The computing time of the pseudo-dynamic assignment is dominated by the chosen number of incremental steps $N$ and $M$. For the first phase a sufficiently large value should be used for $N$ in order to ensure a sufficient independence from the link processing order. For the second phase a coarser stepping is possible. In practical applications values of 5 to 20 for $N$ and values of 2 to 10 for $M$ have been successfully used. If used in practical situations with realistic traffic volumes and realistic network sizes the additional computing time of the proposed simple method is already close to the maximum that would be accepted by users.
4 FIRST EXPERIENCES

The blocking back computation was implemented in the transport planning software package VISUM and is already in use in some projects.

One of the standards in modelling junction dynamics and blocking back is SATURN (Simulation and Assignment of Traffic to Urban Road Networks). The VISUM blocking back mechanism has been compared with SATURN. The following observations were made:

- the resulting queuing pattern is similar to that achieved by VISUM with larger values for N
- dependent on the coding of the junction the blocking back in the case of SATURN may apply to all movements from the incoming link or only to the one directed to the overcapacity successor link.

The latter has been recognised in VISUM through the parameter P for permeability, which determines the extent to which the blocking of one movement at a junction affects the capacity of another. P takes a value between 0 (no interaction) and 1 (full interaction).

The above illustrates that detail in the blocking back applications is desirable, and that therefore run times may be expected to increase. The additional burden is not as in e.g. SATURN which introduces an additional loop in the iterative process, so that a model with x assignment loops and y simulation loops may lead to xy steps in total. In contrast, the blocking back step replaces a standard equilibration step, by recalculting the delays accounting for blocking back effects. However, this still affects calculation times:

a) as the calculations themselves are more involved than the standard steps
b) as the calculations of flow and delay must be reflected in the route balancing process
c) as the introduction of blocking back may destabilise the iterative solution process, leading to slower convergence or possible non-convergence.

The following alternative way was proposed in a first large application project in order to keep the run times manageable whilst also maintaining a converging process:

- calculating the blocking back step not in every assignment step, but only every n\textsuperscript{th} step, with n a decreasing value that seeks to make use of the increasing level of convergence in the assignment process later on in the iterative procedure. Values for n that have been suggested are 10, 8, 5, 5, 5, 2, 1, 1, 1, 1, 1 for a 40-loop assignment
- shifting the volume-delay curves to fit through the corrected (metered) volume and calculated delay (inc blocking back) coordinates, for route balancing purposes.

5 OUTLOOK

It is difficult to position the proposed method somewhere between static and dynamic assignment procedures exactly. The first step away from static assignment is the assumption of a time period during which the "static" situation is valid. Without this assumption, dealing with queues is not possible since in stationary systems queue lengths are infinite in case of oversaturation. The next step in the direction of dynamic assignment models is the incorporation of a rudimentary model of the movement of traffic through the network. All dynamic assignment procedures contain some kind of a flow model in addition to their route choice models. But for being a true dynamic assignment, the proposed method still lacks at least a proper treatment of travel time along the routes; here it sticks with the simplifying assumptions of static assignment procedures.

The authors expect a growing acceptance of truly dynamic assignment methods in practical applications, motivated by the increasing saturation of the infrastructure and the resulting need for a temporally finer resolved and dynamic modeling of transportation processes and congestion phenomena. The development will be complemented by the further improvement of travel demand models and their capability to provide demand data in a high temporal resolution.

However, it will still take some time before dynamic assignment methods have completed their way from academic research to everyday transportation planning practice and have overcome the computing time restrictions by methodological and technical advance. In the meantime, methods like the proposed one provide an attractive compromise.

6 BIBLIOGRAPHY

