Updating turn probabilities at intersections in response to traffic incidents

Sylvie Thiébaux, Peter Lamb, and Bella Robinson
CSIRO Math. & Info. Sc., ITS Project
PO Box 664, Canberra, ACT 2611, Australia
e-mail: `<Firstname>.<Lastname>@cmis.csiro.au`

Abstract

Urban road traffic simulators sometimes require the turn probabilities at the intersections of the network they are simulating. This paper describes a method for updating the given probabilities following the occurrence of a traffic incident. We first estimate the origin and destination of traffic prior to the incident by computing an OD matrix (approximately) consistent with the given turn probabilities and an assignment reflecting the normal traffic conditions. Following the incident, we work backwards from this OD matrix and an assignment reflecting the changing conditions to determine a new appropriate set of turn probabilities. We extend the TRITRAM simulator to simulate the effects of incidents using this method, which we validate on the urban networks constituting TRITRAM's test suite.

Acknowledgements: This research was supported by the Cooperative Research Center for Advanced Computational Systems. Thanks to Garry Francis, Peter Lowrie, Ngoc Nguyen and Doug Quail from RTA Traffic Technology Branch, for their assistance in reviewing an extended version of the paper and suggesting some of the evaluation procedures. Thanks also to Melissa Dobbie and Jeff Wood from CSIRO for their help with statistics.

1 Motivation

Turn probabilities derived from traffic counts can be used as an alternative to origin-destination matrices as inputs to traffic simulation. For small scale traffic simulation (10 to 100 intersections), it may not even be feasible to collect OD matrix data.

The TRITRAM traffic simulator developed by CSIRO and RTA [4], which is the context that motivated this work, requires the conditional turn probabilities at the intersections of the urban road network it is simulating (that is the conditional probability of turning onto a link given the current link). Commonly, these probabilities are gathered at several periods of time representative of typical traffic conditions over a 24 hour period. They therefore model reality adequately under normal circumstances, but are unlikely to do so under abnormal traffic conditions such as those following an incident in the network. In particular, they do not account for the change of route taken by drivers to avoid the congestion caused by the incident, while still reaching their planned destination.

We describe a method for automatically updating these probabilities following an incident. First, the origin and destination of vehicles are determined from the existing probabilities and an assignment reflecting the normal traffic conditions. Then, these probabilities are repeatedly modified so as to account for the changes in the assignment caused by the incident, while still being kept consistent with the estimated OD matrix.

The method has been implemented within the TRITRAM simulator, thereby extending its capabilities to the simulation of traffic incidents. It has been validated on the TRITAM test suite which consists of both artificial and real urban road networks. We found it to be efficient and to provide realistic simulation where the original probability data was realistic.

After an overview of the method (Section 2), we detail the three main tasks involved, namely the computation of an assignment (Section 3), the derivation of the OD matrix (Section 4), and the turn probabilities update itself (Section 5). We then give experimental results obtained with TRITRAM (Section 6), and conclude with some remarks about other problems for which the method could be useful (Section 7). An extended account of this work can be found in [6].

2 Overview of the Method

Let $\mathcal{N}$ be the set of nodes of the network, $\mathcal{O} \subseteq \mathcal{N}$ be the set of origin nodes (input nodes of the network), $\mathcal{D} \subseteq \mathcal{N}$ be the set of destination nodes (sink nodes of the network), $\mathcal{P} = \mathcal{O} \times \mathcal{D}$ be the set of origin-destination pairs, $\mathcal{L}$ be the set of links, and $\mathcal{T} \subseteq \mathcal{L}^2$ be the set of valid turns (pairs of links).

We assume that we know the probability of making any turn $t \in \mathcal{T}$ while transiting through the network under normal traffic conditions. For $t = \langle l_1, l_2 \rangle$, this probability is the average number of times a vehicle travels on link l_1 *and* travels next on link l_2 during the transit. These joint probabilities are what we call the *turn probabilities* in the following. They can easily be determined from traffic counts such as the conditional probabilities mentioned in Section 1 and the flows on the network's input links (and *vice versa*, see [6]). In the following, we note P, the $|\mathcal{T}|$-dimensional column vector whose i^{th} element P_i is the probability of the i^{th} turn. We refer to this vector as the turn probability vector. Our problem is to update the turn probability vector following an incident.

The first step in our approach is to determine for any origin-destination pair $p = \langle o, d \rangle \in \mathcal{P}$, the probability of a vehicle making a trip from o to d. The set of such probabilities is called the *origin-destination* (OD) matrix. In the following, we represent the OD matrix by a $|\mathcal{P}|$-dimensional vector X, whose j^{th} element X_j is the probability of the j^{th} OD pair. The rationale for inferring the OD matrix is that it indicates the planned origin and destination of trips through the network, and that both can be assumed not to have changed following the incident.

What changes however is the route taken by drivers to avoid the congestion caused by the incident. That is, the probability that a trip from o to d makes turn t varies with the current traffic conditions. The set of such probabilities is called an *assignment*. We represent an assignment by a $|\mathcal{T}| \times |\mathcal{P}|$ matrix A, whose $(i,j)^{th}$ element $A_{i,j}$ is the assignment of the trips of the j^{th} OD pair to the i^{th} turn.

Now with these notations, determining the OD matrix X amounts to computing an assignment A reflecting the normal traffic conditions, and solving the linear system of equations

$$AX = P$$

for X. This need only be done once. Updating P following the incident amounts to repeatedly computing a new assignment A reflecting the changing traffic conditions and multiplying it with X.

Implementing this idea requires two main technical choices, for which a range of possibilities exist. The first is that of the model of driver route choice behavior and of a corresponding assignment algorithm. For our implementation within TRITRAM, we chose Dial's multipath proportional assignment model in virtue of its flexibility and small computational complexity [2]. We devised a variant of Dial's algorithm that assigns traffic to the *turns* of the network, based on TRITRAM travel times as a measure of route cost. The second choice to be made is that of an algorithm for solving the linear system of equations. In general, turn probability data is incomplete and inconsistent, so this system is usually both under- and over-determined. Fortunately, there exist algorithms that can provide useful approximate solutions in those circumstances. SMART, the algorithm we use, returns the solution with maximal entropy of the system that requires the fewest unsupported changes to P to be consistent [1]. We now detail these choices in the respective two next sections.

3 Computing the Assignment

Dial's assignment [2] is a proportional multipath assignment, meaning that the proportion of traffic from a given origin to a given destination assigned to a given route is a function of that route's cost (which is independent of the traffic demand), and that more than one route may be assigned some of the traffic. Dial's algorithm was originally designed to assign traffic flows to the network's links. Here we adapt it to assign probabilities to the network's turns. The two main assumptions underlying the model of route choice behavior are that (a) only *reasonable* routes from the origin to the destination may be taken, (i.e., routes that do not contain turns increasing the current cost to the destination or reducing the cost from the origin), and that (b) when faced with two reasonable routes of unequal cost, drivers take the cheapest with a higher probability (reasonable routes of equal costs are equiprobable).

More formally, the cost of a route is taken to be the sum of the costs of the links along this route. In our case, the cost $\mathsf{cost}(l)$ of link $l \in \mathcal{L}$ is the TRITRAM travel time on that link. Let c^* be the cost of the cheapest route from the origin $o \in \mathcal{O}$ to the destination $d \in \mathcal{D}$. To comply with the above assumptions, the probability of taking a reasonable route of cost c linking o to d is defined to be proportional to

$$\exp \theta(c^* - c) \tag{1}$$

for some user-supplied parameter $\theta \in \mathbb{R}^+$. The probability of taking any unreasonable route is defined to be 0. θ is used to control the diversion probabilities, making the model flexible. If $\theta = 0$, then all reasonable routes have the same probability of being taken. When θ increases, the probability of using a cheaper route increases. For large θ, the assignment asymptotically approximates all-or-nothing.

One of the key features of the assignment algorithm is that it is able to assign traffic to all routes according to the probabilities given above, without having to enumerate these routes in any way. The first trick to achieve this is not to work directly with the route probabilities, but rather with the *likelihood* of the *turns* along these routes. For turn $t = \langle l_1, l_2 \rangle \in \mathcal{T}$ the likelihood $h(t)$ is defined as the exponential (times θ) of the difference of cost incurred by going from o to l_2 via l_1 compared to that of the least-cost route from o to l_2. Formally, let $p(l)$ denote the cost of the cheapest route from the origin o to (the start of) link $l \in \mathcal{L}$, and $q(l)$ the cost of the cheapest route from (the start of) l to the destination d. t is part of a reasonable route from o to l_2 iff $p(l_1) < p(l_2)$ and $q(l_2) < q(l_1)$. If this is not the case, taking t leads us to backtrack towards o, or to get further away from d. Now, the cost of the cheapest route from o to l_2 is $p(l_2)$, while the cheapest route going from o to l_2 via l_1 has cost $p(l_1) + \mathsf{cost}(l_1)$. Therefore, the difference of cost incurred by taking t is $p(l_2) - p(l_1) - \mathsf{cost}(l_1)$. Summing up, $h(t)$ is

$$\begin{cases} \exp \theta(p(l_2) - p(l_1) - \mathsf{cost}(l_1)) & \text{if } p(l_1) < p(l_2) \text{ and } q(l_2) < q(l_1) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The utility of this definition is that the probability of a route as defined in equation (1) is proportional to the product of the likelihoods h of the turns along this route. Note that furthermore the probability of a turn is the sum of the probabilities of all routes taking that turn. The first consequence of these facts is if we define $b(l)$ as being proportional to the sum of the probabilities of the routes leading from o to the end of link $l \in \mathcal{L}$, and $a(l)$ as being proportional to the sum of the probabilities of the routes leading from the start of l to d, then the probability of taking turn t when going from o to d is proportional to

$$b(l_1) \times h(\langle l_1, l_2 \rangle) \times a(l_2) \tag{3}$$

```
for $l \in \mathcal{L}$ do
    if $o = \mathsf{origin}(l)$ then $b(l) \leftarrow 1$ else $b(l) \leftarrow 0$

for $l_2 \in \mathcal{L}$ in-increasing-order-of $p$ do
    if $p(l_2) < \infty$ and $b(l_2) = 0$ then
        for $l_1 \in \mathcal{L}$ such that $\langle l_1, l_2 \rangle \in \mathcal{T}$ do
            $b(l_2) \leftarrow b(l_2) + h(\langle l_1, l_2 \rangle) \times b(l_1)$
```

Figure 1: Computation of b via dynamic programming

Since the probabilities of all routes from o to d must sum to 1, the proportionality coefficient K is the inverse of the sum of the $a(l)$ over the links l originating in o.

Another consequence is that a and b can be computed without enumerating routes, using dynamic programming. For computing b, we start by noticing that for all links l having o as origin, the sum of the probabilities of the routes from o to l's origin is 1. That is, $b(l) = 1$ for these links. Now, if we take the other links l in ascending order with respect to their cost $p(l)$ from the origin o, then the sum of the probabilities of the routes from o to the origin of l only depends on the sum of the probabilities of the routes from o to the origin of the links l' turning onto l, and on the likelihoods of the respective turns. That is

$$b(l) = \sum_{\{l' | \langle l', l \rangle \in \mathcal{T}\}} h(\langle l', l \rangle)\, b(l')$$

So b can be computed as in Figure 1. Computing a is similar, see [6].

To sum up, the assignment algorithm consists in the following steps.

1. Based on the cost of a link, compute the cost of the cheapest routes from each link $l_1 \in \mathcal{L}$ to each link $l_2 \in \mathcal{L}$. This can be done for instance using Dijkstra's algorithm on the graph formed by the links of the network [3].
2. For each origin $o \in \mathcal{O}$ (resp. each destination $d \in \mathcal{D}$), compute $p(l)$ wrt. o (resp. $q(l)$ wrt. d) for each link $l \in \mathcal{L}$. This is trivial once step 1. has been completed.
3. For each OD pair $\langle o, d \rangle \in \mathcal{P}$
   (a) Use p and q to compute the likelihood $h(t)$ of each turn $t \in \mathcal{T}$ as indicated in Equation (2).
   (b) Use h to compute $b(l)$ and $a(l)$ for each link $l \in \mathcal{L}$, as indicated in Figure 1.
   (c) Use b, a, and h and to compute the probability of each turn $t \in \mathcal{T}$ as indicated in Equation (3). Multiply by the proportionality coefficient K.
   (d) Report the result in the assignment matrix A.

A fuller treatment including complexity analysis and examples can be found in [6]. Briefly, the total complexity of the algorithm is lower than $O(|\mathcal{L}|^3 \log(|\mathcal{L}|))$, the dominant complexity being that of the dynamic programming algorithm.

4 Computing the OD matrix

Once the assignment matrix A has been computed, we can solve the matrix equation $AX = P$ for the OD matrix X. We now detail the SMART algorithm [1] that we are using for this purpose.

When the matrix equation $AX = P$ has exactly one solution, SMART returns that solution, just as an ordinary method for solving uniquely determined linear systems of equations would.

When the matrix equation admits more than one solution, which is often the case since the number of OD pairs (unknowns) is typically greater than the number of turns (equations), SMART returns the solution with maximal entropy [5]. The rationale behind the maximum entropy solution is that it minimises the implicit information required in addition to that contained in A and P (which is insufficient) for obtaining a unique solution. Selecting any other solution would amount to constraining the available information more than we have to, and therefore to assuming even more information that we do not have. We nevertheless leave the possibility of taking into account more information than A and P if one wishes to, by maximising entropy *relative to a user-provided estimate* of the OD matrix. Such an estimate may be the result of a survey, for instance. So the algorithm takes as input A, P, and a $|\mathcal{P}|$-dimensional vector I, such that element I_j is an estimate of X_j. If no estimate is known at all, I_j may assign 0 to an OD pair $j = \langle o, d \rangle$ for which there is no path between o and d, and 1 otherwise. SMART can be seen as making the fewest unsupported changes to the estimate in order to satisfy the matrix equation. More formally, the entropy of X relative to I is defined as:

$$E(X, I) = -\sum_{j=1}^{|\mathcal{P}|} X_j \log(\frac{X_j}{I_j}) + X_j - I_j$$

and SMART returns the solution of $AX = P$ such that $E(X, I)$ is maximal. Note that if the estimate is already a solution, then its entropy is maximal.

When the matrix equation is inconsistent, meaning that the assumed driver route choice behavior model is inconsistent with the given turn probabilities, then SMART returns the OD matrix with maximal entropy $E(X, I)$ among those OD matrices closest to a solution. X is said to be closest to a solution when it maximises $E(AX, P)$. That is, the algorithm will select the OD matrix with maximal entropy, among the OD matrices that would require the fewest unsupported changes to P in order to be solutions.

Implementing SMART is fairly simple. The algorithm consists in an iteration during which X (initially equal to I) is updated until convergence within a given ϵ is achieved. More precisely, if we note $\bar{X}$ the value of X at the previous iteration step, each element j of X, $1 \leq j \leq |\mathcal{P}|$, is updated as follows:

$$X_j \leftarrow \bar{X}_j \prod_{i=1}^{|\mathcal{T}|} \Big(\frac{P_i}{\sum_{k=1}^{|\mathcal{P}|} A_{i,k} \bar{X}_k} \Big)^{A_{i,j}}$$

network	nodes	links	turns	OD pairs	solution	time (s)
Hong Kong	8	14	18	25	exact	< 0.2
Sydney - Oxford St.	16	29	42	64	exact	0.3
3-Grid equip. turns	21	49	108	144	approximate	0.5
3-Grid	21	49	108	144	approximate	0.5
Sydney - George St.	74	148	211	676	approximate	3.1
Albury	74	184	291	1295	approximate	8.5

Table 1: SMART run-times

with the convention that $X_j = 0$ whenever $\bar{X}_j = 0$, and that if the denominator $\sum_{k=1}^{|\mathcal{P}|} A_{i,k}\bar{X}_k$ equals 0, then the fraction equals 1. Provided an implementation of A as a sparse matrix, the complexity of an iteration step is proportional to the number of non-zero elements in A.

The algorithm always converge, either to a solution if there is one, or else to an approximate solution. In our implementation, we chose to stop iterating when the two-norm of the change in X is smaller than ϵ, i.e., $\sqrt{\sum_{j=1}^{|\mathcal{P}|}(X_j - \bar{X}_j)^2} \leq \epsilon$. We are not aware of any theoretical result concerning the rate of convergence of SMART, which is still a current research topic [1], but for the needs of our application, we were satisfied with SMART's performance (see results below).

5 Updating the turn probabilities

The OD matrix X need only be calculated once at the start of the simulation. Following an incident, we update the turn probability vector P as follows. At regular time intervals, we recompute the assignment matrix A, using the current link travel times from the simulation as the link costs. Multiplying A with X gives us an update of P which is then used for the next simulation period, and so on. This suffices to simulate the effects of incidents on the traffic flows.

6 Experimental Results

We extended our C^{++} implementation of TRITRAM to simulate the effects of incidents using this approach, and ran it on a Sparc Ultra 1 with 64 Mb of memory. We conducted experiments on a test suite comprising real networks of areas of Hong Kong, Syndey, and Albury, as well as artificial grid networks. For the latter, we considered the case where the probabilities of going straight through (p_s), turning right (p_r), and turning left (p_l) at each intersection were equal, and the case where $p_s = 0.7$, $p_l = 0.2$, and $p_r = 0.1$. The size of these networks is given in Table 1.

Our first experiment confirmed that SMART run-times are acceptable. As can be seen in Table 1, only networks of the size of Albury start to be challenging.

We next looked at the type of solution return by SMART. For two first networks, SMART finds an exact solution to the equation $P = AX$ (see Table 1), showing the compatibility of Dial's route choice behavior model and the turn probability data. This implies that if we multiply the assignment matrix A with the OD matrix X

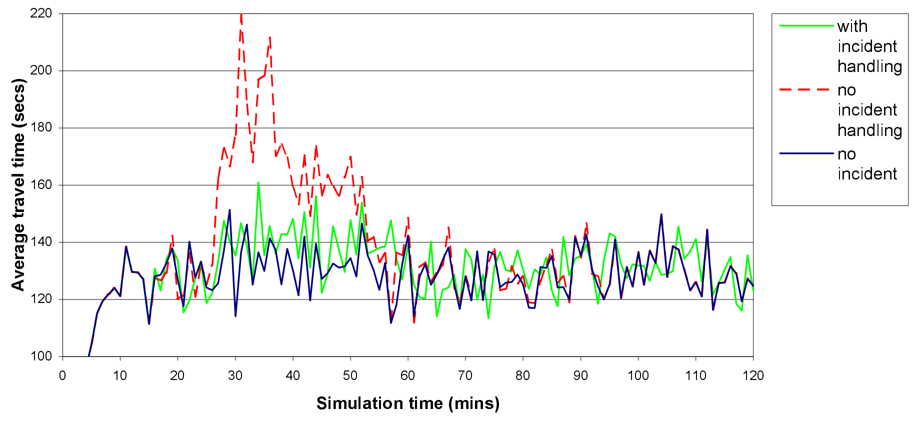

Figure 2: Incident simulation with and without turn probability update

returned by SMART, we get our original probability vector P back. In the other cases, we only get an approximation $\tilde{P}$ of P. By comparing $\tilde{P}$ to P, we found that the quality of the approximation depends on the quality of the data. For the grid network with equiprobable turns, the probabilities are nearly consistent and the approximation never exceeds 0.2%. For the 3 other networks, the probability data is less realistic or accurate, and the approximation is usually within a few percent. However, when missing probabilities are made up, the difference between the corresponding components of $\tilde{P}$ and P can be arbitrarily large.

Even in the case where the matrix equation is consistent, it rarely has a unique solution because the information contained in the turn probabilities is insufficient to determine a unique OD matrix. To determine to what extent this constitutes an obstacle to our approach, we randomly generated 50 OD matrices for the Oxford Street network, determined the corresponding turn probabilities by multiplying, in each case, the assignment with the OD matrix, and looked at whether or not our approach could re-discover the original OD matrix from the turn probabilities and the assignment. Since our ultimate goal is to update the turn probabilities following a change in traffic conditions, we also compared the updated turn probabilities obtained from the original and reconstructed OD matrices, respectively, following a sample traffic incident. We found that the distance between the original and reconstructed OD matrices was quite large, 35% of the total flow being incorrectly recovered on average. Fortunately, this has very little impact on the turn probabilities, for which we only observed an average shift of 0.01% of the total flow.

Figure 2 shows the results of simulation of the 3-Grid network in TRITRAM where an incident blocked 2 of 3 lanes of a link from t=15 to t=25. The average travel time through the network with and without turn probability update can be compared with the simulation run without incident. Without turn probability update, the travel time is severely affected by links filling with vehicles prevented from entering the link blocked by the incident. With the update, traffic is routed around the blocked link, and although the travel time increases, it does not do so excessively. The experiment used link travel time from the simulation as the link cost, and $theta \simeq 0.14\text{min}^{-1}$, which decreases the link usage to 50% of its initial value if the link cost increases by 50%.

7 Conclusion

We have presented an approach to the update of existing traffic counts following a change of traffic conditions. This approach attempts to construct a plausible OD matrix reflecting the available data using Dial's model of driver route choice behavior. The probabilities are then updated by making them consistent with this OD matrix and an assignment reflecting the new traffic conditions.

In addition to answering our initial problem, this method addresses several other interesting issues as a by-product. We can test the existing traffic counts against a reasonable driver route choice behavior model. It suffices to check whether the matrix equation is solvable. We can extract an OD-matrix from possibly incomplete traffic counts. Indeed, this constitutes the first step of our approach. Although this paper has not exploited the fact that the matrix equation could well have less equations than the number of turns, the approach allows for turn counts to be missing. When the traffic counts are highly incomplete or when their information content is weak, it might be best to use them to improve an OD matrix estimate rather than to compute the OD matrix from scratch. This also can be achieved within our framework. The approach can also be used to infer complete turn probabilities from the values of the flows and partial turn counts. Indeed, this amounts to computing an OD matrix consistent with the available data, and to inferring the turn probabilities from this OD matrix and the assignment.

We therefore believe that this work could be profitably applied in other contexts such as split plan generation, to provide a complete set of turn probabilities from vehicle detected counts and, optionally, turn survey data.

References

[1] C. Byrne. Convergent block-iterative algorithms for image reconstruction from inconsistent data. *IEEE Transactions on Image Processing*, 6(9):1296–1304, 1997.

[2] R. B. Dial. A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research*, 5:83–111, 1971.

[3] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerical Math*, pages 269–271, 1959.

[4] V. N. Nguyen, P. Kilby, and P. Lamb. Validation of the TRITRAM traffic network simulation model. In *Proc. 3rd International Conference of ITS Australia (ITSA-97)*, Brisbane, Australia, 1997.

[5] C. E. Shannon. A mathematical theory of communication. *Bell System Technology Journal*, 27:379–423, 623–656, 1948.

[6] S. Thiébaux and P. Lamb. Updating turn probabilities at intersections in response to traffic incidents. Technical Report CMIS 98/140, CSIRO, 1998.