ROThAr: Real-time On-line Traffic Assignment with Load Estimation

Takfarinas Saber*†, Anthony Ventresque* and John Murphy*
*Lero@UCD, School of Computer Science and Informatics, University College Dublin, Ireland
Email: {anthony.ventresque,j.murphy}@ucd.ie
† University of Nantes, France
Email: takfarinassaber@gmail.com

***Abstract*—More and more drivers use on-board units to help them navigate in the increasing urbanised environment they live and work in. These system (e.g., routing applications on smart phones) are now very often *on-line*, and use information from the traffic situation (e.g., accidents, congestion) to get the best route. We can now envisage a world where all trips are assigned and updated by such an on-line system, making the best routing decisions based on traffic conditions. The problem is that current systems consider only 'local' elements (e.g., driver preference and current traffic condition) and do not make routing decisions from a global perspective. This can lead to a lot of similar routing assignments that could lead to further traffic congestion. The objective of the next generation on-line navigation systems is then to come up with a 'smart', real-time route assignment, which balances the load between the different road segments and offers the best quality to the drivers. However, every routing decision made has an impact on the traffic conditions (one more vehicle on the road segments selected) and computing the load induced by the trips is a computationally heavy problem. This paper addresses this question of real-time on-line traffic assignment, and shows that under certain conditions it is possible to have (i) an accurate estimation of the load and travel time on every road segment and (ii) an optimised traffic assignment that adapts to divergence and evolutions (e.g., accidents) of the system.**

***Keywords*-Global Real-time and On-line Routing; Load and Travel Time Estimation; Traffic Assignment.**

I. Introduction

According to the United Nations Population Fund (UNFPA), cities are much more attractive than rural areas: they create more jobs and income opportunities, have better life standards, etc. That is the reason why half of the world's population live in urban areas and this number will increase to reach almost 5 Billion by 2030 [1], i.e. 58% of the world population. This is already the case for up to 76% of the population in the European region [2]. And on top of that, we should add the also important and growing urban commuters population – i.e., people living in rural areas but working and spending most of their time in cities.

This situation stresses the transport systems of urban areas to unprecedented levels and forces decision makers to take actions.

For instance, in the greater Dublin area (GDA) the commuters using their car increased from 46.7% in 1996 to 51.8% in 2006, while the proportion of primary school students travelling as car passengers increased from 29.5% in 1996 to 46.9% in 2006 [3]. Unfortunately, these two categories share the same time window for their trips to work/school and increase the level of congestion on the roads. This has a well-known negative impact on the users and the infrastructures: safety, productivity of employees, etc., all undergo from time wasted in transport. On the other hand, the money invested in public transports keeps flowing and reached €211 million in 2011 for the same GDA [4]. In times where every cent counts, public authorities have to find solutions that offer the best possible service and lower the costs.

The first solution to overcome these issues dates back the late 70s and early 80s with traffic management systems (TMS) such as the Sydney Coordinated Adaptive Traffic System (SCATS) [5] and others. These systems collect information directly from the road network using various sensors and try to optimise the traffic flows. Sensors can be present in the roads, such as induction loops which are able to count the number of vehicles and track their speed. They are linked to the transport regulation elements (e.g., traffic lights, LED traffic signs) and try to tune the flows of vehicles to avoid congestion or react to particular events.

Whatever the positive impact of these systems on the transport systems is, they do not cover all areas, especially since their deployment and maintenance is expensive: small cities cannot afford them, while rural areas do not need such complex infrastructure. Moreover, they tend to have blind spots, i.e., wherever they have a good coverage they do not give a full picture of the traffic situation. Another more and more popular method for collecting data is to use the digital footprints, i.e., all metadata that users of digital devices generate when they use their phones or their navigation devices [6]. This can allow a TMS to get the location of users and/or cars, their speed, and possibly their destination.

All this information can be helpful for route planning, and is a leap forward compared to classical air communication (e.g., radio, TMC[1]). Drivers have two kinds of services for planning their individual trips: off-line (e.g., TomTom) with the route planning capabilities self-contained – and any update needs to be uploaded from the Internet – and *on-line* (e.g., Google maps) when the navigation service is accessed through the

[1]Traffic Message Channel, a technology that enables traffic info to be sent to on-board units.

Internet and resides on a central server. With the advent of Internet-enabled devices (e.g., using 3G) the on-line model seems to be the most popular, and we can easily expect it to be predominant in the near future.

It seems we can start to think of a group of on-line routing providers collaborating under a public authority to *assign the best route* to all the drivers. This seems to fall under the general question of *traffic assignment*, i.e., giving a route to every vehicle in a transportation setting [7], [8], [9]. Our problem is a little different though as we look for the best routes for every trips in a transportation system that match users' needs, minimise their travel time, and provide the routes on real-time. In short: *can we find a real-time (for it has to be useful for drivers), on-line (because it is the only way to follow the individual assignments) traffic assignment solution that finds and maintain optimised routes?*

In this paper, we introduce ROThAr[2], a Real-time On-line Traffic Assignment system that (i) routes vehicles according to some routing algorithm (ROThAr is agnostic to the routing algorithms); (ii) evaluates the travel time on the road segments and updates it after every assignment, which allows it to predict where to route trips; and (iii) tries to optimise the traffic conditions by rerouting vehicles, using the estimated travel times.

The rest of this paper is structured as follows. Section II defines the problem and the quality of a traffic assignment. Section III gives an overview of the system and aims at giving a sense of how ROThAr works. Section IV describes the load and travel time estimation module. Section V shows how the rerouting and system optimisation module works. We show some evaluations in section VI and finally we discuss our approach and conclude in section VII.

II. Problem Definition

Sometimes also called *Route Assignment* or *Route Choice*, *Traffic Assignment* (TA) is the main element of the fourth and last stage in the *Conventional Transportation Forecasting Model* [10]. Needless to say it is important for transportation infrastructure design and planning. This problem consists in selecting a route $r(v)$ for every vehicle v in the system, given an origin $o(v)$ and a destination $d(v)$ for the vehicle.

In this section we present some general definitions first, and then we formulate what kind of TA we address.

A. General Definitions

Definition 1 (Road Network): A *road network* is a directed multigraph $G(N, S, c)$ where $N = \{n_i, \ldots, n_m\}$, $m \in \mathbb{N}$, is a set of nodes denoting notable road elements: junctions, ends, etc., $S = \{s_1, \ldots, s_o\}$, $o \in \mathbb{N}$, $\forall s_i, \exists (n_j, n_k), s_i = (n_j, n_k)$ is a set of arcs (directed edges) representing road segments between nodes, and c is a function assigning some properties to each arc and denoting some information about it[3].

Every driver in the system selects a route for each of their trips. Let's define the concepts of *trip* and *route* first:

[2]Rothar means bicycle in Irish language.

[3]For simplicity, we do not use this information in this paper.

Definition 2 (Trip): A *trip* $\mathcal{T} \in T$ is a tuple $\mathcal{T}(v) = < o(v), t_{o(v)}, d(v), p(v) >$, where v is a vehicle, $o(v)$ its origin, $t_{o(v)}$ the starting time of its trip, $d(v)$ its destination, and $p(v)$ a set of preference regarding the best route for v's driver (e.g., time, toll costs).

Definition 3 (Route): A *route* $r \in R$ is a list of road segments: $r = \{s_0, \ldots, s_j\}$, $j \in \mathbb{N}$. A route aims at satisfying a given trip $\mathcal{T}$, and in this case we have $\exists n_k, n_l \in N, s_0 = (o(v), n_k)$ and $s_j = (n_l, d(v))$.

The definition of TA below is rather simple and embraces the ideas developed by others [7], [8], [9]. In short, TA consists in assigning a route for every trip in the system.

Definition 4 (Traffic Assignment): A *Traffic Assignment* (TA) is a function that gives a route $r \in R$ for every trip $\mathcal{T} \in T$. $TA : T \mapsto R$ such that $TA(\mathcal{T}) \to r$.

B. Traffic Flow Models

Traditional TA [11] consists in defining routes for all the trips of a population of drivers and a road network, and to evaluate/predict the situation and bottlenecks. It is used by infrastructure designers and decision makers to plan evolutions and modifications of the system – long-, mid- or even short-term.

A large amount of work has been done in the area of traffic flow models for modelling traffic using either trip-based models (e.g., static or dynamic models) [12], [13], [14], activity based models, and economics models. Other models address traffic propagation (e.g., macro-, micro- or meso-models) [15], [16], [17]. This field of study is not really the purpose of this paper and we do not present these discussions here – we refer the reader to the literature referenced.

In this paper, we do not try to model the flows and their dynamics, but to adapt the routes to the traffic situation and to (re-)assign routes to trips. So of course we share some concerns with the traffic flow models (specially the static and dynamic traffic assignments [12], [14]) but we have our own definition of the problem.

C. Real-time On-line Traffic Assignment

Definition 5 (Real-time On-line Traffic Assignment): A *Real-time On-line Traffic Assignment* ROTA is a function that gives a route $r \in R$ for every trip $t \in T$. $ROTA : T \mapsto R$ such that $ROTA(\mathcal{T}) \to r$. One difference with a TA is that a ROTA keeps optimising the utility function of the system: $\forall t_i, t_j, t_i \neq t_j TA^{t_i}(\mathcal{T}) = TA^{t_j}(\mathcal{T})$ while $ROTA^{t_i}(\mathcal{T})$ may be different than $ROTA^{t_j}(\mathcal{T})$. Another difference is that the processing time of a ROTA has to be bounded by ϵ, ϵ being the time to process the assignment (e.g., 20 seconds).

III. Overview of our Solution: ROThAr

Our solution is called ROThAr, for Real-time On-line Traffic Assignment with load estimation. It is composed of three modules that work together to come up with an optimised TA (see Figure 1).

Users give their original position, destination and time of departure to the on-line routing module. This module uses the

global knowledge about the road traffic conditions given by the traffic condition estimation module. The module queries and updates a structure called $\mathcal{LM}$: see Section IV. The route optimisation module is triggered by some events happening to the traffic conditions (e.g., accidents) to modify the routes. It can also run on a regular basis (e.g., every 5 minutes) to adapt the system to small divergence on the traffic situation and so on.

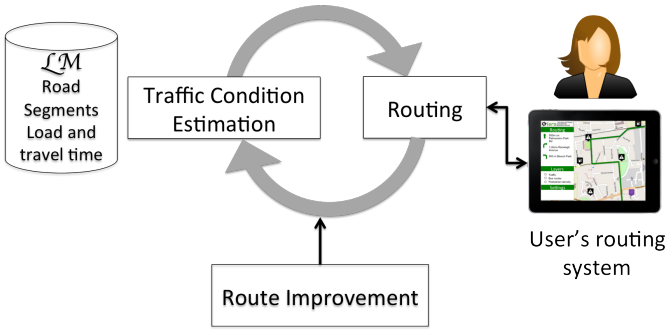


Figure 1. ROThAr system overview.

A. Routing

The routing module is out of the scope of this paper. Any classical routing algorithm can be used as long as it can handle the information provided by $\mathcal{LM}$.

B. Estimation of Road Segments Travel Time and Load

This module is the core of our system as it is the one which is supposed to offer ROThAr with an accurate view on the traffic condition and its evolution during the assignments. We show in Section IV that computing the position of vehicles accurately is unrealistic: traffic is too stochastic and any minor variation can lead to large differences in what would be the real position of vehicles. For instance, reaction time at traffic lights varies greatly between drivers, and that has an obvious impact on the flows of vehicles. Besides, computing the position of every vehicle and their impact on load and travel time on road segments seems heavy. In this paper we propose to split the time in similar time slots, and to give an estimated position of vehicles at every time slot. Obviously, the smaller the time slot, the more accurate the position (and the most expensive the computation of vehicles positions).

C. Rerouting and Traffic Assignment Optimisation

Once routes have been assigned to all vehicles, ROThAr tries to optimise the traffic flows by rerouting some of the vehicles. We will see later that it is an action which is undertaken only if time permits – i.e., this does not have an impact on routing features of the system, especially as we want a real-time routing/rerouting. The rerouting component iterates on the routes as long as it does not take too much time: ROThAr takes some critical trips (e.g., those which create too much load on some segments) and try to assign them better routes (e.g., avoiding overloaded segments); Section V describes the details of this module.

IV. $\mathcal{LM}$: Estimation of Road Segments Travel Time and Load

The challenge here is to get an accurate and efficient estimation of the position of every vehicle over the time, given the other vehicles in the road-network.

A. Travel time

Travel time of a trip is usually considered an additive function of the time spent on each road segments composing the trip [18]. It looks like an over-simplification, as one can expect that the stochastic nature of transportation slows down or speeds up things at junctions, pedestrian crossings, etc. But it is assumed that this composition of trip elements (road segments) gives a rather good estimation of the travel time. For each road segment, the speed of a vehicle is impacted by many factors, among which we have the speed limit, the characteristics of the road (e.g., slope, shape and visibility) and the number of other vehicles on this segment (load). Speed limit and characteristics of the road are static variable, and do not need to be evaluated at run-time. On the other hand, the load evolves continually and keeps impacting the average speed.

Travel time on a segment s_i is classically given by the *Bureau of Public Roads* (BPR) [18]'s formula:

$$tt(s_i, l(s_i)) = tt(s_i, 0) \times \left[1 + \alpha_{s_i} \left(\frac{l(s_i)}{C(s_i)}\right)^{\beta_{s_i}}\right] \quad (1)$$

where $tt(s_i, 0)$ is the *free-flow* travel time, $l(s_i)$ and $C(s_i)$ respectively the load and the capacity of the segment s_i. The constants α_{s_i} and β_{s_i} are BPR technical constants related to s_i ($\alpha_{s_i} > 0$, $\beta_{s_i} \geq 1$).

The main issue we find with this formula is that the load is not bounded by the capacity of the segment s_i, while, in reality, when the maximum capacity is reached, there cannot be any more vehicles on the road. We then propose a slight modification of the formula to take this element into account: in our formula below, when the load $l(s_i)$ on segment s_i reaches the capacity $C(s_i)$, we cap the travel time – and no more vehicle can be added to the road segment.

$$tt(s_i, l(s_i)) = \begin{cases} tt(s_i, 0) \times \left[1 + \alpha_{s_i} \left(\frac{l(s_i)}{C(s_i)}\right)^{\beta_{s_i}}\right] \\ \qquad\qquad \text{if } l(s_i) \leq C(s_i) \\ tt(s_i, 0) \times [1 + \alpha_{a_i}] \quad \text{otherwise} \end{cases} \quad (2)$$

B. Road Segments Load Estimation at Every Time Slot

Estimating the load on a segment, i.e., the number of vehicles that are present on this segment at any given time t seems a big challenge as the arrival time of vehicles, their order on the road, and their number, all have an impact on the load and its evolution, and also on the complexity of the computation. The best solution for this problem is probably to use a microscopic simulation, such as SUMO [19], VISSIM [20], MATSim [21] and FLoMiTSiM [22]. This traffic model allows a simulation at the individual vehicle level and gives a precise view of the system. But these techniques are expensive

and we do not believe it would be possible to use them for real-time traffic assignment in the medium or large scale urban area network that we are targeting.

Instead, our idea is based on providing a set of segments of roads where the vehicle v may be present, for every time unit of a certain size $\mathcal{U}$.

1) Computing the Load for a Single Time Slot: If we estimate that vehicle v is in the segment of road s_i, at the i^{th} position of its route $r(v) = \{s_0, \ldots, s_i, \ldots, s_m\}$, $m \in \mathbb{N}$, and this at $u \times \mathcal{U}$ seconds, then the set of segments for which the load is increased in $\mathcal{LM}$ is:

$$toBeLoaded(v, i, u) = (s_i, s_{i+1}, \ldots, s_k) \tag{3}$$

$$s.t. \begin{cases} \sum_{j=i}^{k} tt(s_j, 0) \leq \mathcal{U} \wedge k = m \\ \text{or} \\ \sum_{j=i}^{k-1} tt(s_j, 0) \leq \mathcal{U} \wedge \sum_{j=i}^{k} tt(s_j, 0) > \mathcal{U} \end{cases}$$

For each vehicle v active in the system, a load of $+1$ is added to all segments in the set $toBeLoaded(v, i, u)$, but only for this u^{th} time slot.

2) Overload Propagation: All the traffic flow models we came across manage the traffic jams, (i.e., when the load on the road segment s_i at the time slot u $l(s_i, u)$ is greater than or equal to the capacity of this one) with a vehicle buffer associated to the entrance of s_i. We believe that travel time on segments leading to s_i has to be impacted also.

$$\forall s_i, s_j \in S, s_i = (n_k, n_l), s_j = (n_p, n_q), l(s_i, u) > C(s_i),$$
$$n_q = n_k, tt(s_j, l(s_j, u)) = tt(s_j, 0) \times \left[1 + \alpha_{s_j}\right]$$

Note that this impact on the travel time does not mean any impact on load: if s_i is congested, the travel time on s_j is set to its maximum – vehicles cannot exit the segment easily, while the load stills the same. This makes sure that there is no 'snowball effect', as load is not directly propagated to s_j.

3) Repositioning: Once the load for each s_i in the time slot u has been computed and the propagation of the overload to s_j done, we get the position of every vehicle.

$\forall v \in V, \mathcal{T}(v) = < o(v), t_{o(v)}, d(v), p(v) >, r(v) = \{s_0 = o(v), \ldots, s_i, \ldots, s_m = d(v)\}$, $m \in \mathbb{N}$, the position of v at the time slot $u + 1$ can be expressed according to its position s_i at the time slot u:

$$\begin{cases} If\ u \times \mathcal{U} \leq t_o(v) & pos(v, u) = 0 \\ \text{else} & pos(v, u+1) = \{ \begin{array}{ll} k & \text{if } \exists k \\ m & \text{otherwise} \end{array} \end{cases} \tag{4}$$

k is defined as follows:

$$\begin{cases} \sum_{i=pos(v,u)}^{k} tt(s_i, l(s_i, u)) \geq \mathcal{U} \\ \text{and} \\ \sum_{i=pos(v,u)}^{k-1} tt(s_i, l(s_i, u)) < \mathcal{U} \end{cases}$$

with $tt(s_i, l(s_i, u))$, the travel time on the segment s_i at the u^{th} time slot. The issue is here that the travel time on a single segment s_i can be bigger than $\mathcal{U}$, either because of the load on s_i, $l(s_i, u)$, or because $\mathcal{U}$ is too small. In this case, the vehicle would be stuck in an infinite loop, if ROThAr only uses the formulas above. Two simple options can be considered to fix this issue: (i) track precisely the position of each vehicle on a segment or (ii) always move the vehicles by at least one segment after every time unit. The first one is computationally expensive, while second one is less realistic.

ROThAr implements a third option: when $\mathcal{U}$ is too small for a segment s_i, v is penalised by a certain number of time units where it has to stay in s_i before going to the next one in $r(v)$:

$$penalty(v, s_i) = \begin{cases} \lceil \frac{tt(s_i, l(s_i, u))}{\mathcal{U}} \rceil & \text{if } \mathcal{U} < tt(s_i, l(s_i, u)) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

4) Evolution by time slot: The process of computing $\mathcal{LM}$ is rather simple:

- As long as there are vehicles such that $s_{pos(v,u)} \neq d(v)$, a new entry $u + 1$ is added to $\mathcal{LM}$.
- We update the $\mathcal{LM}$ for every vehicle. We propagate congestion and we reposition vehicles according to corresponding travel time.

Algorithm 1 shows the pseudo code for the computation of loads matrix:

The variation of the time-unit induces a variation in the precision and the execution time. When $\mathcal{U}$ tends to the infinity, loads can be computed in one iteration, but it is considered that a vehicle induces a load on each segment of road that it crosses as it is the case in the STA based on the flow. Conversely, when $\mathcal{U}$ tends to zero, the algorithm needs more iterations. This time-unit obliges vehicles to stay more time-units in the same arcs. The result is that the loads matrix has more time slots and its computation is more expensive.

V. (RE-)ROUTING

Our objective is to optimise the traffic conditions by giving a good route to new vehicles and improving the route of vehicles already in the system. The search space of the problem we are addressing here (large road network, thousands of trips) is huge and we target a real-time computation of the routing/rerouting. It is then important to apply efficient and non-expensive heuristics that find at least one solution at the end of the time allowed to the improvement. A trade-off has to be found between the quality of the improvements and the computation time, in order to find at least one solution, with some improvement from the previous state, in real-time. Real-time can be interpreted here as "in less than 20 seconds": this seems a small enough amount of time for the rerouting to happens, and it is similar to what already exists in classical routing systems.

A. Computation of the Matrix of Loads $\mathcal{LM}$

This is a crucial element of the system, as it is used as input to the routing algorithms and can make the difference between a routing and a good routing: one which sends all vehicles to the same, soon overloaded segments, and one which does not. This matrix is computed by algorithm 1 and it is based on

Algorithm 1: Computation of $\mathcal{LM}$

input : Road network: $\mathcal{G}(N, S, c)$, list of vehicles $v \in \mathcal{V}$ with their routes $r(v) \in \mathcal{R}$, time-unit $\mathcal{U}$
output: up-to-date $\mathcal{LM}$

$\mathcal{LM} = Matrix()$;
// vehicles begin their routes
$vehiclePosition = [1, 1, \ldots, 1]$;
// penalty: v stays $n \times \mathcal{U}$ in s_i
$vehicleInJam = [0, 0, \ldots, 0]$;
$u = 0$;
while $\exists v_i, pos(v_i, u) \neq d(v_i)$ **do**
  // Compute load
  for $s_i \in \{1, \ldots, |S|\}$ **do**
    $\mathcal{LM}[s_i][u] = 0$;
  $\mathcal{V}^{\overline{D}} \leftarrow getVehiclesNotInDestination()$;
  $S^{OL} \leftarrow \emptyset$;
  for $v_i \in \mathcal{V}^{\overline{D}}$ **do**
    $oldPosition \leftarrow vehiclePosition[v.index]$;
    $S^L \leftarrow toBeLoaded(v, oldPosition, u)$;
    for $s \in S^L$ **do**
      $\mathcal{LM}[s.position][u] \mathrel{+}= 1$;
      if $s.isOverloadedAt(u)$ **then**
        $S^{OL} \bigcup= \{s\}$;
  // Overloads propagation
  for $s \in S^{OL}$ **do**
    $node = s.originNode()$;
    $S^{In} = node.getInSegments()$;
    for $impactedS \in S^{In}$ **do**
      $newLoad \leftarrow impactedS.getCapacity()$;
      $\mathcal{LM}[impactedS.position][u] \leftarrow newLoad$;
  // Repositioning
  for $v_i \in \mathcal{V}^{\overline{D}}$ **do**
    if *vehicleInJam[v.index] = 0* **then**
      $oldPosition \leftarrow vehiclePosition[v.index]$;
      $newPosition \leftarrow pos_{v,u}(oldPosition)$;
      $vehiclePosition[v.index] \leftarrow newPosition$;
      if *oldPosition = newPosition* **then**
        $slotsInJam \leftarrow \left\lceil \frac{\mathcal{LM}[oldPosition][u]}{\mathcal{U}} \right\rceil$;
        $vehicleInJam[v.index] \leftarrow slotsInJam - 1$;
      else
        $vehicleInJam[v.index] \mathrel{-}= 1$;
        if *vehicleInJam[v.index] = 0* **then**
          $vehiclePosition[v.index] \mathrel{+}= 1$;
  $u \mathrel{+}= 1$;
return $\mathcal{LM}$;

vehicles that are already in the system (for which a route has been assigned to).

The main difficulty is the picking of an adequate time unit for the computation of this matrix: a long time unit implies less precision while a short one slows down the computation of the entire matrix. We do a validation of this trade-off in Section VI.

B. Routing Vehicles as They Arrive

The first important thing that any routing system should provide, and that includes our ROThAr, is that it finds a route for every new vehicle starting its journey. In ROThAr, a new route is created for every new v_i using a $Dijkstra$ shortest path. ROThAr obviously leverages on the load matrix $\mathcal{LM}$ to compute a route that minimises the travel time: for every road segment $s_j \in r(v_i)$, v_i arrives at s_j at an estimated time slot u depending on the travel time of all segments before s_j in $r(v_i)$. If the load of s_j at this time slot is $l(s_j, u)$, then the cost that corresponds to selecting it is:

$$Cost_{s_j} = tt(s_j, L{+}1) + L \times [tt(s_j, L+1) - tt(s_j, L)] \quad (6)$$

such that: $L = l(s_j, u)$.

The equation 6 expresses the fact that selecting a road-segment has not only an impact on the vehicle that we want to route, but also on the others that have already planned to go through it.

Like $Dijkstra$ algorithm applied in a graph with fixed costs (not changing with the time), the routing of each vehicle has a quadratic time complexity, but based on the number of arcs and not on the number of nodes. This is due to the fact that we consider that the position of a vehicle is always in a segment of road and not in a junction.

At the end of the algorithm, a load of one is added to each segment of roads s_j that composes the route $r(v_i)$ at adequate time slot.

C. Improvement

The improvement of the traffic condition is the final part of our system. It is done for as much time as we can afford, given the previous steps, and operates as follows:

- At each iteration, we look for the most congested segment of road, say s_i. The idea is to re-route some vehicles, in order to avoid this area, reduce load on s_i and incidentally reduce the travel time on s_i.
- We reroute first the vehicles that will go through s_i, but are not too close yet to it. This is because it is easier to reroute trips when they are not immediately in contact with s_i, and also because otherwise the algorithm would load segments very close to s_i, resulting in just a translation of the problem to the region around s_i.
- For similar reasons, our system does not reroute vehicles having a destination $D(v)$ too close to s_i.
- Eventually, the matrix of load $\mathcal{LM}$ is updated according to the new routes given to vehicles.

A *tabu list* of a certain size k is used for the selection of the most overloaded arc. It works as a queue, where the selected

Algorithm 2: Improvement

input : Road network: $\mathcal{G}(N, S, c)$, $\mathcal{LM}$, list of vehicles $v \in \mathcal{V}$ with their routes $r(v) \in \mathcal{R}$.
output: List of vehicles $v \in \mathcal{V}$ with their new routes, up-to-date $\mathcal{LM}$.

$tabuList \leftarrow \emptyset$;
while *Enough time* **do**
  $s \leftarrow overloadedSegment(\mathcal{LM}, tabuList)$;
  $\mathcal{V}^I \leftarrow impactedByOverLoad(\mathcal{V}, s)$;
  $\mathcal{V}^I \setminus = inSameArea(\mathcal{G}, \mathcal{V}, s)$;
  $\mathcal{V}^I \setminus = goingToSameArea(\mathcal{G}, \mathcal{V}, s)$;
  for $v \in \mathcal{V}^I$ **do**
    $\mathcal{LM}.substractLoad(v.route)$;
    $adaptedDijkstra(\mathcal{G}, \mathcal{LM}, v)$;
    $\mathcal{LM}.addLoad(v.route)$;
return $\mathcal{V}$, $\mathcal{LM}$;

arc is conserved during k iterations before removing it. This is in order to avoid the selection of the same arc during all the improvements, and at the same time allow its re-improvement after a while.

VI. Experimental results

We have sought a medium size road network for our experiments, taken from a real environment in order to demonstrate that our solution can work and scale. We have extracted the road network of the island of Manhattan in New-York from OpenStreetMap [23], composed of 412 nodes (junctions) and 799 directed arcs (one-way roads).

The travel time on each arc is defined by Formula 2 and we use the following values for the variable:

- the free flow travel time on every segment s_i is set to the maximum speed on a urban area road: $tt(s_i, 0) = 50km/h$[4].
- the capacity of each lane is defined by its length divided by the average vehicle size ($4m$), and most roads in Manhattan have two lanes, therefore $\forall s_i C(s_i) = 2 \times \lfloor \frac{roadLength}{vehicleSize} \rfloor$.
- Finally, $\beta_{s_i} = 4$ (see [24] for more details) and $\alpha_{s_i} = 5$. All our experiments are developed in *Python 2.7.3* on an Intel ® Core™ $i7-2600$ CPU @ $3.40GHz$ and $16GB$ of RAM machine running Debian GNU/Linux 7.0 (wheezy) $x86\_64$.

A. Execution time of $\mathcal{LM}$

The first thing we evaluate is the execution time of the matrix computation. It consists of evaluating the load (and incidentally the travel time) of each segment for every time unit for as long as there are vehicles planned to be on the segments. Every time a new trip $\mathcal{T}$ is reported, it has potentially an impact on the whole matrix.
We consider 14 different possible time units: {6s; 12s; 18s; 24s; 30s; 60s; 120s; 240s; 360s; 600s; 1,200s; 3,600s; 7,200s; ∞} and 9 possible load in the network (number of vehicles): {100; 200; 500; 1,000; 2,000; 5,000; 10,000; 50,000; 100,000}. We believe they cover enough possible options in terms of time unit and number of vehicles. For every vehicle v we create a trip $\mathcal{T}(v)$ (origin and destination are set randomly using a uniform function) and we fetch a route $r(v)$ using a Dijkstra shortest path algorithm. Tests are executed six times for each, and an average execution time is reported here in Figure 2.

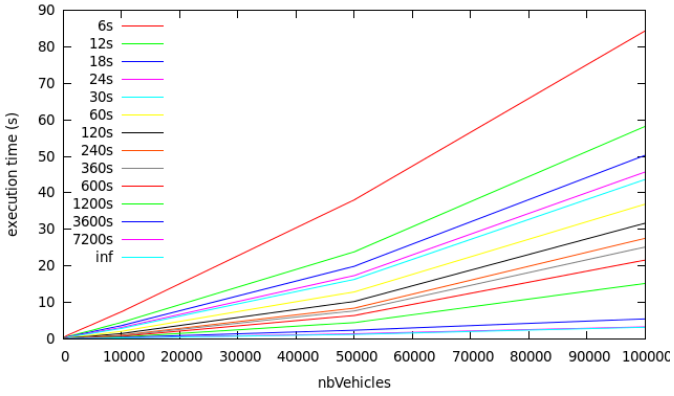


Figure 2. Execution times of $\mathcal{LM}$ according to the time units and number of trips.

Figure 2 shows three distinct things: (i) that the execution time grows linearly with the number of vehicles, which is interesting as it allows to predict the execution time of $\mathcal{LM}$ and the most appropriate time unit depending on the size of the problem, the computation power of the machine; (ii) that execution time is inversely proportional to the time unit size: the lower the time unit, the bigger the gradient of the increase of the execution time; (iii) that the two biggest time units (in this example 7,200s and ∞) have exactly the same values: this because 2h (7,200s) looks like the longest travel time, i.e., the travel time for a vehicle on the 'diameter' of the road network (longest path); bigger time slots do not capture more information therefore.

B. Improvement(s) and Real-time

It is difficult to have an estimation of what real-time means in our scenario, especially since it may vary depending on the vehicles – emergency cars may need a faster re-routing than random drivers stuck in a traffic jam... We choose the value under which the re-routing can be considered real-time at *20s*. In this second set of experiments, we evaluate the number of new trips that get a route (likely to be the most important value) and the number of iterations (each of them being to some

[4]It is a little unrealistic as some roads have a different value and in average the real speed is a little slower, but it would not change the overall comparison of different approaches.

extent an optimisation step of the traffic conditions). We use the same set-ups as previous experiments but add $X\%$ new vehicles after the initial computation of $\mathcal{LM}$: ROThAr then tries to give all these new vehicles a route and to improve the conditions, by rerouting some vehicles – until the 20s window is reached. Note that we have more vehicles than in the previous experiment: $(1 + ratio) \times |nbVehicles|$.

Figures 3 show the number of routing and rerouting: a positive value means that all trips ('old' and 'new') are assigned a route and the system gets improved (some trips get rerouted) while a negative value means some trips could not get a route. We see that the number of improvements decreases with the number of (old) vehicles already in the network until reaching 0 between 10,000 and 50,000 vehicles for most experiments. Besides, the bigger the time slot (e.g., compare Figures 3 (a) and (c)) the bigger the improvement. The key fact is that the ratio of new vehicles added in the system does not seem to have a strong impact on the number of improvements. This is partially due to the quick routing itself (simple Dijkstra with random origins and destinations on a medium road network), as well as the $\mathcal{LM}$ update. The difference then lies in the iterations and the size of the system.

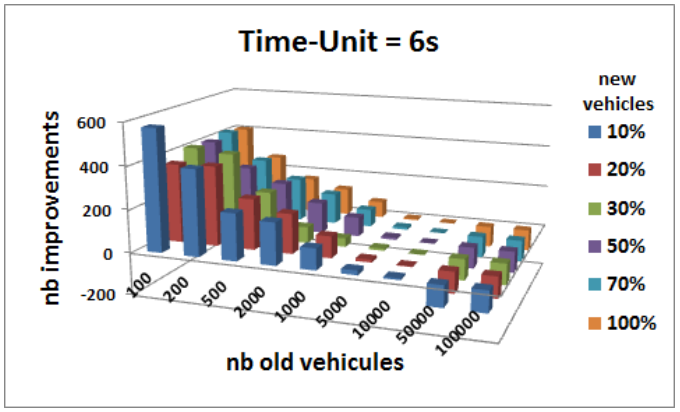
C. Improvement of the Travel Time

The last evaluation relates to the travel time experienced by the drivers. This is probably the only one which really matters for the users – how much they spend during their trips. We use the exact same set-ups as previous experiments and compare their results to a simple Dijkstra-like trip planner. We call 'old' the vehicles that have been routed already (and may benefit from the improvements) and 'new' the vehicles that have to be routed first and then potentially rerouted (see previous Section VI-B). 'Global' is a term that covers both 'new' and 'old'. We use three scenarios with, respectively, 1,000, 5,000 and 10,000 vehicles, a 6s time unit and a ratio of 50% new vehicles – so in total we have 1,500, 7,500 and 15,000 vehicles in the simulation.

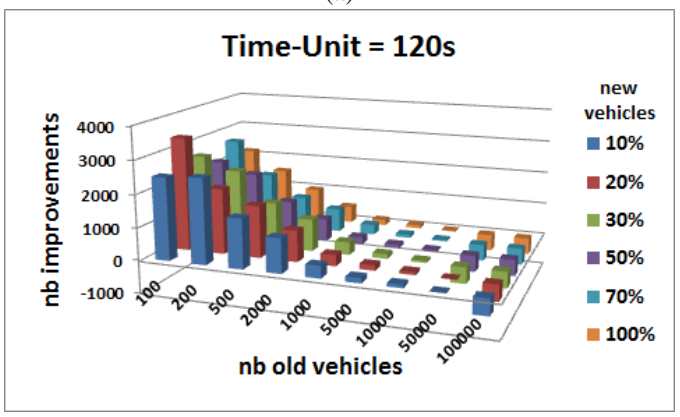
Table I shows the average percentage and value of saved travel time for each type of vehicle (old, new and global). The more vehicles there are, the smaller is the average improvement. We can also see that new vehicles benefit more than old ones of the improvements. That makes sense as ROThAr chooses a good routing *and* tries to improve the routes of all vehicles – double optimisation to some extent. Eventually, while the average saved travel time seems small (41.72s for 15,000 vehicles), this corresponds to more than 173 hours for the entire network.

In next two sets of experiments, we want to analyse the population of cars in their relation to the travel time optimisation: how many see a smaller travel time? How many see their situation worsening?

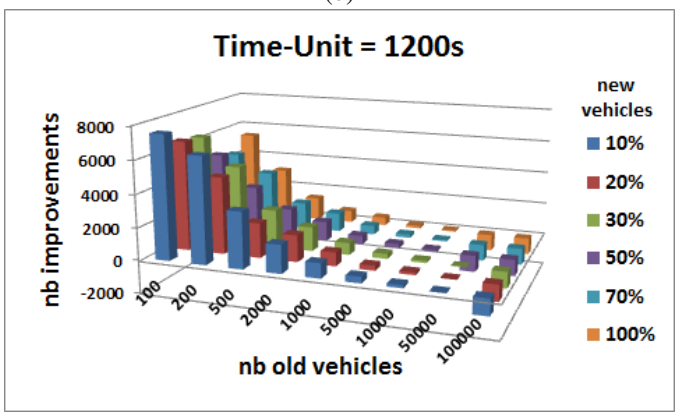
First, we look at the vehicles that improve their travel time. Table II shows that even in the medium scenario



(a)



(b)



(c)

Figure 3. Number of improvements (routing/re-routing) with time units 6s, 120s and 1,200s. Note that y-axes scales are different.

Table I
GLOBAL IMPROVEMENT (SAVED TRAVEL TIME).

nb old vehicles	old	new	global
1000	1.80% 144.0797s	3.95% 366.6854s	2.52% 218.2816s
5000	0.95% 81.3993s	1.10% 86.4799s	1.00% 83.0928s
10000	0.54% 40.2595s	0.64% 44.6505s	0.55% 41.7232s

(10,000 vehicles) 9.36% of the vehicles experience a better travel time – less for the old vehicles than for the new ones. Of course, as they are 10% only, travel time saved for them is one order of magnitude better for them than it is in Table I.

Table II
IMPROVEMENT IN TRAVEL TIME AS A RESULT OF PROPOSED NEW ROUTES (RATIO OF VEHICLES AND AVERAGE TIME SAVED).

nb old vehicles	old	new	global
1000	28.00% 656.9295s	43.60% 945.0028s	33.20% 783.0339s
5000	15.18% 611.5081s	16.96% 610.0003s	15.77% 610.9677s
10000	8.88% 490.3066s	10.32% 499.6467s	9.36% 493.7393s

Finally, Table III shows that only a few vehicles experience more delays with ROThAr, but the situation can be bad for them (almost 5 minutes extra delays in average for them).

Table III
DEGRADATION IN TRAVEL TIME AS A RESULT OF PROPOSED NEW ROUTES (RATIO OF VEHICLES AND AVERAGE TIME WASTED).

nb old vehicles	old	new	global
1000	7.80% -511.0323s	8.20% -552.8751s	7.93% -525.4487s
5000	2.30% -496.8548s	3.32% -511.3301s	2.64% -502.9227s
10000	0.75% -437.2952s	2.56% -270.0398s	1.35% -331.8337

VII. CONCLUSION AND FUTURE WORK

In this paper, we described two components of modern traffic assignment: (i) a road segment travel time (or load) estimation matrix and (ii) a rerouting algorithm. Both constitute the core of ROThAr, a novel real-time on-line traffic assignment solution. We show that ROThAr increases the global utility function of the system: on average vehicles spend less time travelling. We also demonstrated that the computation of the travel time matrix is tractable and facilitates routing improvements for a subset of the vehicles – especially as the approximation of their position becomes less exact.

We believe, from the literature survey we made during this research, that our approach is seminal. In particular, the problem we define, namely *real-time, on-line traffic assignment*, has not previously been proposed in the literature, despite it being a pertinent problem in the traffic management domain. Likewise, the load matrix appears to be, to the best of our knowledge, a novel solution for road segment travel estimation. It provides a unique global overview of the traffic conditions to the routing algorithms. We anticipate in future work that we can further utilise the matrix to gain additional improvement in routing.

ACKNOWLEDGEMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

REFERENCES

[1] "United nations population fund (unfpa)," http://www.unfpa.org/pds/urbanization.htm,.

[2] "World bank - urbanization," http://data.worldbank.org/topic/urban-development,.

[3] N. Commins and A. Nolan, "The determinants of mode of transport to work in the greater dublin area," *Transport Policy*, vol. 18, no. 1, pp. 259 – 268, 2011.

[4] "National transport authority," http://www.nationaltransport.ie/projects-schemes/transport-projects/,.

[5] A. G. Sims and K. Dobinson, "The sydney coordinated adaptive traffic (scat) system philosophy and benefits," *Vehicular Technology, IEEE Transactions on*, vol. 29, no. 2, pp. 130–137, 1980.

[6] F. Calabrese, M. Diao, G. D. Lorenzo, J. F. Jr., and C. Ratti, "Understanding individual mobility patterns from urban sensing data: A mobile phone trace example," *Transportation Research Part C: Emerging Technologies*, vol. 26, no. 0, pp. 301 – 313, 2013.

[7] J. G. Wardrop, "Some theoretical aspects of road traffic research," in *Institute of Civil Engineers*, vol. 1, no. 3, 1952, pp. 325–378.

[8] M. Beckmann, C. McGuire, and C. B. Winsten, "Studies in the economics of transportation," Tech. Rep., 1956.

[9] S. P. Equilibrium, T. N. Equilibrium, O. M. Equilibrium, W. P. Equilibrium, and F. Equilibrium, "Network economics: A variational inequality approach," *Advances in Computational Economics*, vol. 1, 1993.

[10] M. G. McNally, "The four step model," 2008.

[11] S. Maerivoet, "Modelling traffic on motorways: State of the art, numerical data analysis, and dynamic traffic assignment," Ph.D. dissertation, Katholieke Universiteit Leuven, Jun. 2006.

[12] D. Boyce, "Forecasting travel on congested urban transportation networks: review and prospects for network equilibrium models," *Networks and Spatial Economics*, vol. 7, no. 2, pp. 99–128, 2007.

[13] W. Jastrzebski, "Volume delay functions," in *15th international EMME/2 Users' Group Conference. Vencouver BC: BPRW Planowanie Projektowanie Doradztwo SA*, vol. 9, 2000.

[14] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, "Dynamic traffic assignment: A primer," *Transportation Research E-Circular*, no. E-C153, 2011.

[15] M. Papageorgiou, "Some remarks on macroscopic traffic flow modelling," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 5, pp. 323–329, 1998.

[16] M. Brackstone and M. McDonald, "Car-following: a historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999.

[17] S. P. Hoogendoorn and P. H. Bovy, "State-of-the-art of vehicular traffic flow modelling," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 215, no. 4, pp. 283–303, 2001.

[18] T. A. Manual, "Urban planning division," *US Department of Commerce, Washington DC*, 1964.

[19] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, October 2011, pp. 63–68.

[20] M. Fellendorf, "Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *64th Institute of Transportation Engineers Annual Meeting*, 1994, pp. 1–9.

[21] J. W. Joubert, P. J. Fourie, and K. W. Axhausen, "Large-scale agent-based combined traffic simulation of private cars and commercial vehicles," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2168, no. 1, pp. 24–32, 2010.

[22] M. Errampalli, M. Okushima, and T. Akiyama, "Development of the microscopic traffic simulation model with the fuzzy logic technique," *Simulation*, vol. 89, no. 1, pp. 87–101, 2013.

[23] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
[24] A. Raith, J. Y. Wang, M. Ehrgott, and S. A. Mitchell, "Solving multi-objective traffic assignment," *Annals of Operations Research*, pp. 1–34, 2011.