Location of turning ratio and flow sensors for flow reconstruction in large traffic networks

Martin Rodriguez-Vega, Carlos Canudas de Wit, Hassen Fourati

Location of turning ratio and flow sensors for flow reconstruction in large traffic networks

Martin Rodriguez-Vega[a,*], Carlos Canudas-de-Wit[a], Hassen Fourati[a]

[a] *CNRS, Gipsa-Lab, Grenoble INP, Univ. Grenoble Alpes, INRIA, 38000 Grenoble, France*

Abstract

In this work we examine the problem of minimizing the number of sensors needed to completely recover the vehicular flow in a steady state traffic network. We consider two possible sensor technologies: one that allows the measurement of turning ratios at a given intersection and the other that directly measures the flow in a road. We formulate an optimization problem that finds the optimal location of both types of sensors, such that a minimum number is required. To solve this problem, we propose a method that relies on the structure of the underlying graph, which has a quasi-linear computational complexity, resulting in less computing time when compared to other works in the literature. We evaluate our results using dynamical traffic simulations in synthetic networks.

Keywords: Sensor location, Flow estimation, Large traffic networks

1. Introduction

Traffic state monitoring and estimation play a key role in the development of intelligent transportation systems. Because of the high installation and maintenance cost of sensors, it is of great importance to efficiently locate the least amount of sensors that give enough information about the network state. Fortunately, as different sensing technologies start to become available at a smaller cost, heterogeneous data can be incorporated in a cooperative way in order to optimize the information while minimizing the sensing locations.

In this paper, we consider the use of two types of sensing technologies to reconstruct the state of the traffic network. Flow sensors are installed in road segments and return the number of vehicles that passed over the location during a period of time. On the other hand, turning ratio sensors are installed on intersections and can estimate the percentage of vehicles turning to any of the exits of the junction. In contrast to flow sensors, turning ratio sensors are required to measure routing choices, and not necessarily total flow. For instance, Abbott-jard et al. (2013) and Bhaskar et al. (2015) studied the use of Bluetooth and WiFi technology to obtain vehicle trajectories, and by identifying entry/exit

*Corresponding author. E-mail address: Martin.Rodriguez-Vega@gipsa-lab.fr

road pairs in intersections, routing proportions can be estimated. This strategy does not provide the total flow since the penetration rate is usually unknown. A different strategy is the use of data recollection campaigns with mobile flow sensors to construct a database of average routing behavior. Other methods were briefly discussed by Bianco et al. (2001).

In the literature, different approaches have been considered for the determination of the minimum number of sensors according to the degree of knowledge of the network and the available sensing technologies. Bianco et al. (2001) defined the region of influence of flow sensors located at intersections, and with the knowledge of turning ratios in the network, they were able to provide necessary conditions for flow observability and some heuristic bounds on the number of sensors. Bianco et al. (2006) performed a deeper analysis and determined that the sensor location problem is NP-complete.

Some methods, developed by Castillo et al. (2008) and Hu et al. (2009) formulated the problem by enumerating all possible paths of the network to construct a matrix of constraints to give bounds on the required number of sensors to achieve flow reconstruction. Furthermore, they analyzed the effect of different network topologies on these bounds. Additionally, if there is available information about unused paths, Castillo et al. (2013) and Castillo et al. (2014) provided extensions to reduce the number of possibilities that need to be enumerated. Hu and Liou (2014) and Fu et al. (2016) used these methods to consider the use of not only stationary flow sensors, but also active sensors capable of providing floating car data.

A different alternative was presented by Ng (2012), who considered the conservation of flow equations at intersections instead of path flows. This method requires no enumeration of paths and is able to calculate the number of sensors as a function of the number of nodes and edges in a graph. Additionally, they proposed an algorithm for flow sensor location. This method was improved by He (2013), who provided an efficient algorithm based on the topology of the network and graph theory. However, as this approach uses little assumptions about the network (only topology is used), the number of sensors may be large for real applications Viti et al. (2014).

More robust approaches to this problem have also been studied. He (2013), Viti et al. (2014) and Rinaldi and Viti (2017) proposed methods to assess and locate sensors for partial observability, i.e. only some of the flows of the network are reconstructed. Other works such as Xu et al. (2016) and Lovisari et al. (2016) deal with measurement noise presenting a trade-off between the number of sensors and the quality of flow reconstruction. More complete reviews about models and methods used for flow reconstruction and sensor location can be found in Gentili and Mirchandani (2012) and Castillo et al. (2015).

Our contribution in this paper consists in solving the sensor location problem by considering flow and turning ratio sensors. This allows for a cost trade-off between these sensing technologies which can reduce the total number of sensors. The method only relies on knowledge of network topology, and does not require any knowledge about route choices or O/D paths which may be hard to obtain in certain cases. Additionally, we noted while studying the literature that most

Table 1: Basic notations used in this work

Notation	Description
$\mathbb{I}$	Identity Matrix
$\mathbb{1}$	Vector of ones
$\mathbf{0}$	Vector of zeros
$\mathbf{u}_i$	Standard basis vector with 1 in the i-th coordinate.
$\lvert\mathcal{A}\rvert$ or $n_{\mathcal{A}}$	Cardinality of set $\mathcal{A}$.
$\mathcal{A} \setminus \mathcal{B}$	Relative complement of sets $\mathcal{A}$ and $\mathcal{B}$.
$\ker A$	Null space (or kernel) of matrix A
$e_i \in \mathcal{A}$	i-th element of set $\mathcal{A}$
$A_{i,j}$	Element of matrix A at row i and column j.
$\mathbf{v}(j)$	j-th component of vector $\mathbf{v}$.

of the proposed methods present a high computational complexity. We propose sensor location algorithms that allow for complete flow reconstruction with the minimal number of sensors, with quasi-linear complexity with respect to the number of intersections and roads. Thus, these algorithms can be used for very large traffic networks.

2. Notation and definitions

In this paper, calligraphic letters $\mathcal{A}$ are used for sets, uppercase letters A are used for matrices, boldface lowercase letters $\mathbf{a}$ are used for vectors, and lowercase letters a are used for scalars. Some basic notations used throughout this work are shown in Table 1.

We represent traffic networks by means of a directed graph. The nodes of the graph are partitioned in two disjoint sets $\mathcal{C}$ and $\mathcal{N}$: $\mathcal{C}$ corresponds to source and sink nodes of the network and $\mathcal{N} = \{1, 2, \ldots, n_{\mathcal{N}}\}$ represents intersections which are not able to generate or store vehicular flow. The edges $\mathcal{E} = \{1, 2, \ldots, n_{\mathcal{E}}\}$ represent the set of roads of the network. Denote $\mathcal{I}(k)$ as the set of incoming edges to some node k and $\mathcal{O}(k)$ as the set of outgoing edges from k. We define a partition of $\mathcal{E}$ in three sets: $\mathcal{E}_{\text{in}} = \bigcup_{k\in\mathcal{C}} \mathcal{O}(k)$ are the boundary incoming roads, $\mathcal{E}_{\text{out}} = \bigcup_{k\in\mathcal{C}} \mathcal{I}(k)$ are the boundary outgoing roads, and $\mathcal{E}_{\text{net}} = \mathcal{E} \setminus (\mathcal{E}_{\text{in}} \cup \mathcal{E}_{\text{out}})$ are the internal roads of the network.

Definition 1. A feasible traffic network is a directed graph $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ such that the following conditions are met:

- Every edge is part of a path that starts with an edge from $\mathcal{E}_{\text{in}}$ and ends with an edge from $\mathcal{E}_{\text{out}}$.
- The graph contains no self loops.
- $\forall k \in \mathcal{N}$ we have $\mathcal{I}(k) \cap \mathcal{E}_{\text{out}} = \varnothing$ and $\mathcal{O}(k) \cap \mathcal{E}_{\text{in}} = \varnothing$.
- There is no production or storage of vehicles in the nodes in $\mathcal{N}$.

Definition 2. A turning ratio $r_{i,j}$ is the percentage of vehicular flow in road $i \in \mathcal{E}$ that turns to road $j \in \mathcal{E}$ at some intersection $k \in \mathcal{N}$ for which $i \in \mathcal{I}(k)$ and $j \in \mathcal{O}(k)$. □

From now on if two roads i and j are not connected to the same node as indicated in Definition 2, we denote $r_{i,j} = 0$.

3. Problem statement

Our problem is to locate flow and turning ratio sensors such that the road flows (also known as link or edge flows) of the network can be inferred from the measurements while minimizing the number of sensors. In the literature this is known as the *network sensor location problem*, Hu et al. (2009, 2016). We point out that we do not recover path or origin-destination flows in this paper.

To formulate this problem, we must first define the linear constraints that arise from the flow conservation equations.

3.1. Flow conservation equations at intersections

As we are considering steady-state, each road $e \in \mathcal{E}$ is characterized by a unique vehicular flow φ_e. From the definition of the turning ratios (Definition 2), each of the outgoing flows of any intersection $k \in \mathcal{N}$ can be expressed as the sum of the contributions of the incoming flows weighted by the corresponding turning ratio:

$$\varphi_j - \sum_{i \in \mathcal{I}(k)} r_{i,j}\varphi_i = 0 \quad , \quad \forall j \in \mathcal{O}(k). \tag{1}$$

Additionally, the flow conservation requires that the total incoming flow must be equal to the total outgoing flow:

$$\sum_{j \in \mathcal{O}(k)} \varphi_j - \sum_{i \in \mathcal{I}(k)} \varphi_i = 0. \tag{2}$$

Note that these two equations imply that $\sum_j r_{i,j} = 1$.

It can be seen that eq. (1) can only be used when the turning ratios are known. Let $\mathcal{R} \subseteq \mathcal{N}$ be the set of intersections equipped with turning ratio sensors. For each of the outgoing edges of the intersections in $\mathcal{R}$, an equation following (1) can be written. Define matrix $A(\mathcal{R})$ as the collection of the resulting equations for all intersections in $\mathcal{R}$:

$$A(\mathcal{R})_{i,j} = \begin{cases} 1 & \text{if} \quad e_i = j \\ -r_{j,e_i} & \text{else} \end{cases} , \tag{3}$$

where e_i is the i-th element of the ordered set $\bigcup_{k \in \mathcal{R}} \mathcal{O}(k)$. As an edge can only belong to one source node, it is clear that the sets $\mathcal{O}(k)$ for all $k \in \mathcal{N}$ are mutually disjoint. Therefore, the number of rows of $A(\mathcal{R})$ is $|\bigcup_{k \in \mathcal{R}} \mathcal{O}(k)| = \sum_{k \in \mathcal{R}} \deg^{\text{out}}(k)$, where $\deg^{\text{out}}(k)$ is the out-degree (number of outgoing edges) of node k.

Let $\mathcal{U} = \mathcal{N} \setminus \mathcal{R}$ be the set of unmeasured intersections. Following a similar discussion to $\mathcal{R}$, each of the intersections in $\mathcal{U}$ has assigned an equation analogue to (2). Define matrix $B(\mathcal{U})$ as the collection of resulting equations for each $k \in \mathcal{U}$:

$$B(\mathcal{U})_{i,j} = \begin{cases} 1 & \text{if} \quad j \in \mathcal{O}(k_i) \\ -1 & \text{if} \quad j \in \mathcal{I}(k_i) \\ 0 & \text{else} \end{cases}, \tag{4}$$

where k_i is the i-th element of the ordered set $\mathcal{U}$. Check that $B(\mathcal{U})$ has size $(n_{\mathcal{N}} - n_{\mathcal{R}}) \times n_{\mathcal{E}}$ with $n_{\mathcal{R}} = |\mathcal{R}|$.

3.2. System of linear constraints

To represent the location of flow sensors, let $\mathcal{S} \subseteq \mathcal{E}$ be the set of roads with cardinality $|\mathcal{S}| = n_s$ equipped with flow sensors. We define a matrix $C(\mathcal{S}) \in \{0,1\}^{n_s \times n_{\mathcal{E}}}$ such that $C(\mathcal{S})_{i,j} = 1$ if the i-th sensor is located in the j-th road. Note that we can write $C(\mathcal{S}) = [\mathbf{u}_{e_1}\ \mathbf{u}_{e_2}\ \cdots\ \mathbf{u}_{e_{n_{\mathcal{S}}}}]^T$, where e_i is the i-th element of the set $\mathcal{S}$. We assume that the elements of $\mathcal{S}$ are distinct, and so rank $C(\mathcal{S}) = n_{\mathcal{S}}$.

Using these matrices, we can write the linear constraints on the system.

$$\begin{bmatrix} L(\mathcal{R}) \\ C(\mathcal{S}) \end{bmatrix} \boldsymbol{\varphi} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\varphi}_m \end{bmatrix}, \tag{5}$$

where $L(\mathcal{R}) = [A(\mathcal{R})^T\ B(\mathcal{U})^T]^T$ and $\boldsymbol{\varphi}_m$ are the measurements given by the flow sensors.

Example 1. Consider the traffic network shown in Figure 1. Denote $\mathcal{C} = \{v^+, v^-\}$ where v^+ is a source node and v^- is a sink node. Additionally, the intersections are denoted by $\mathcal{N} = \{1, 2, \ldots, 6\}$ and the roads by $\mathcal{E} = \{1, 2, \ldots, 11\}$. Assume that turning ratio sensors are located at intersection $\mathcal{R} = \{2, 3\}$, and thus $\mathcal{U} = \{1, 4, 5, 6\}$. Applying the previous description, the linear constraint

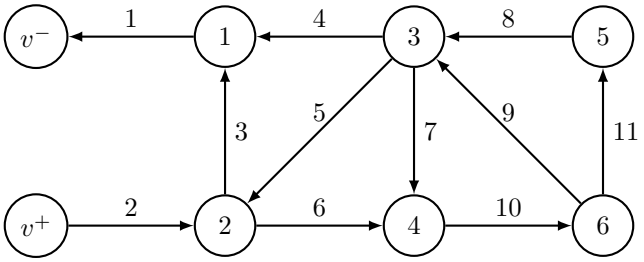

Figure 1: Simple traffic network.

matrices are

$$A(\mathcal{R}) = \begin{array}{c} \\ \varphi_3 \\ \varphi_6 \\ \varphi_4 \\ \varphi_5 \\ \varphi_7 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 & \varphi_5 & \varphi_6 & \varphi_7 & \varphi_8 & \varphi_9 & \varphi_{10} & \varphi_{11} \end{array} \\ \begin{bmatrix} 0 & -r_{23} & 1 & 0 & -r_{53} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -r_{26} & 0 & 0 & -r_{56} & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -r_{84} & -r_{94} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -r_{85} & -r_{95} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -r_{87} & -r_{97} & 0 & 0 \end{bmatrix} \end{array},$$

$$B(\mathcal{U}) = \begin{array}{c} \\ 1 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 & \varphi_5 & \varphi_6 & \varphi_7 & \varphi_8 & \varphi_9 & \varphi_{10} & \varphi_{11} \end{array} \\ \begin{bmatrix} 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix} \end{array}.$$

The first two rows of $A(\mathcal{R})$ correspond to the outgoing edges of intersection 2 (roads 3 and 6), and the last three rows are related to the outgoing edges of intersection 3 (roads 4, 5 and 7). Similarly, each of the rows of $B(\mathcal{U})$ are associated with intersections 1, 4, 5 and 6. Now consider that sensors are located at edges $\mathcal{S} = \{1, 9\}$. The resulting sensing matrix is

$$C(\mathcal{S}) = \begin{array}{c} \begin{array}{ccccccccccc} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 & \varphi_5 & \varphi_6 & \varphi_7 & \varphi_8 & \varphi_9 & \varphi_{10} & \varphi_{11} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

Note that the first row of $C(\mathcal{S})$ is $\mathbf{u}_1^T$ and the second one is $\mathbf{u}_9^T$.

As a specific case, let the turning ratios be such that the input flows 2, 5, 8 and 9 are distributed evenly to their corresponding destinations. It is straightforward to check that

$$\operatorname{rank}\left(\begin{bmatrix} A(\mathcal{R}) \\ B(\mathcal{U}) \\ C(\mathcal{S}) \end{bmatrix}\right) = 11.$$

Therefore, this matrix is invertible and flow reconstruction can be done. □

It is clear that the system (5) has a solution when the matrix in the left hand side has rank equal to the number of columns (the number of roads in the network $n_{\mathcal{E}}$). Therefore, the condition for full road flow observability is

$$\operatorname{rank}\left(\begin{bmatrix} L(\mathcal{R}) \\ C(\mathcal{S}) \end{bmatrix}\right) = n_{\mathcal{E}}. \tag{6}$$

3.3. Network sensor location problem

Our objective is to find the minimum number of flow and turning ratio sensors and their positions. For the moment, we will consider the scenario where the number of turning ratio sensors $|\mathcal{R}|$ is a known number $n_{\mathcal{R}}$ and the number of flow sensors $|\mathcal{S}|$ is minimized. Later, in Section 6, we will expand the results to the general case where both $|\mathcal{R}|$ and $|\mathcal{S}|$ are the decision variables.

The problem formulation is to find the optimal sets $\mathcal{R}^*$ and $\mathcal{S}^*$ that solve

$$\begin{aligned} \mathcal{R}^*, \mathcal{S}^* \quad = \quad & \underset{\mathcal{R},\mathcal{S}}{\text{argmin}} \quad |\mathcal{S}| \\ & \text{subject to} \quad \text{rank} \left(\begin{bmatrix} L(\mathcal{R}) \\ C(\mathcal{S}) \end{bmatrix} \right) = n_{\mathcal{E}} \ , \\ & \qquad\qquad |\mathcal{R}| = n_{\mathcal{R}} \end{aligned} \tag{7}$$

where $n_{\mathcal{R}}$ is a given number of intersections to be equipped with turning ratio sensors.

Proposition 1. *For any $n_{\mathcal{R}} \in \{0, 1, \ldots, n_{\mathcal{N}}\}$, problem (7) has at least one feasible solution.*

Proof. Consider $\mathcal{S} = \mathcal{E}$ for which $C(\mathcal{E}) = \mathbb{I}_{n_{\mathcal{E}}}$. It is straightforward that $\text{rank} \left(\begin{bmatrix} L(\mathcal{R}) \\ C(\mathcal{E}) \end{bmatrix} \right) = n_{\mathcal{E}}$ for any matrix $L(\mathcal{R})$. This selection of sets $\mathcal{R}, \mathcal{S}$ satisfies the constraints, hence the feasible set is non empty. □

It can be seen that this problem is combinatorial. Using a brute force approach, i.e. calculate the cost for every possible combination and selecting the optimal one, would require $O((n_{\mathcal{N}}/n_{\mathcal{R}})^{n_{\mathcal{R}}}\, 2^{n_{\mathcal{E}}})$ operations, which is expensive to solve. Because of this, we consider a partition of this problem into two problems with a lower degree of complexity. The first consists in independently calculating the optimal set of turning ratio sensors as

$$\begin{aligned} \mathcal{R}^* \quad = \quad & \underset{\mathcal{R}}{\text{argmax}} \quad \text{rank } L(\mathcal{R}) \\ & \text{subject to} \quad |\mathcal{R}| = n_{\mathcal{R}} \end{aligned} \ . \tag{8}$$

Once we find the set $\mathcal{R}^*$, the second problem consists in finding any set $\mathcal{S}^*$ that satisfies the following conditions:

$$\begin{aligned} & \text{find any} \quad \mathcal{S}^* \subseteq \mathcal{E} \\ & \text{such that} \quad \text{rank } C(\mathcal{S}^*) = n_{\mathcal{E}} - \text{rank } L(\mathcal{R}^*) \\ & \qquad\qquad \text{rank} \left(\begin{bmatrix} L(\mathcal{R}^*) \\ C(\mathcal{S}^*) \end{bmatrix} \right) = \text{rank } C(\mathcal{S}^*) + \text{rank } L(\mathcal{R}^*) \end{aligned} \ . \tag{9}$$

Next, we show that solving problems (8) and (9) is equivalent to solving problem (7).

Lemma 1. *For any given set $\mathcal{R}^* \subseteq \mathcal{N}$, problem (9) has at least one solution.*

Proof. From the definition of $L(\mathcal{R}^*)$, it can be seen that

$$0 < \text{rank } L(\mathcal{R}^*) \leq \sum_{k \in \mathcal{N}} \deg^{\text{out}}(k) < n_{\mathcal{E}}.$$

Consider $\mathcal{S} = \mathcal{E}$ so $C(\mathcal{E}) = \mathbb{I}$, and rank $C(\mathcal{E}) = n_{\mathcal{E}}$. It follows that rank $C(\mathcal{E})$ + rank $L(\mathcal{R}^*) > n_{\mathcal{E}}$. However, rank $\left(\begin{bmatrix} L(\mathcal{R}^*) \\ C(\mathcal{E}) \end{bmatrix} \right) = n_{\mathcal{E}}$, so some rows of $C(\mathcal{E})$ are linear combinations of the other rows. We can find a solution with the following iterative process:

1. Initialize $\mathcal{S} = \mathcal{E}$.
2. While rank $L(\mathcal{R}^*)$ + rank $C(\mathcal{S}) > n_{\mathcal{E}}$
   2.1 Find $s \in \mathcal{S}$ such that the corresponding row of $C(\mathcal{S})$ is linear combination of the other rows of the matrix.
   2.2 Assign $\mathcal{S} \leftarrow \mathcal{S} \setminus \{s\}$
3. Define $\mathcal{S}^* \leftarrow \mathcal{S}$

As the process only removes redundant rows, the rank of the augmented matrix is still equal to $n_{\mathcal{E}}$. Additionally, the algorithm only ends when rank $L(\mathcal{R}^*)$+ rank $C(\mathcal{S}) = n_{\mathcal{E}}$. Therefore, for a given $\mathcal{R}^*$, it is always possible to find an $\mathcal{S}^*$ which satisfies the constraints. □

Lemma 2. *Let $\mathcal{R}^*, \mathcal{S}^*$ be an optimal solution to problem* (7)*. Then, the rows of matrices $L(\mathcal{R}^*)$ and $C(\mathcal{S}^*)$ are linearly independent.*

Proof. We proceed by contradiction. Assume that the rows of $C(\mathcal{S}^*)$ are not linearly independent to the rows of $L(\mathcal{R}^*)$. There exist at least one row of $C(\mathcal{S}^*)$, $\mathbf{u}_s^T$ such that $s \in \mathcal{S}^*$, which is a linear combination of the rows of $\begin{bmatrix} L(\mathcal{R}^*) \\ C(\mathcal{S}^*) \end{bmatrix}$.

Define a new set $\mathcal{S}' = \mathcal{S}^* \setminus \{s\}$. Matrix $C(\mathcal{S}')$ is the same as $C(\mathcal{S}^*)$ but with row $\mathbf{u}_s^T$ removed. It is evident that rank $\left(\begin{bmatrix} L(\mathcal{R}^*) \\ C(\mathcal{S}') \end{bmatrix} \right) = n_{\mathcal{E}}$ so the constraints are still satisfied. Additionally, $|\mathcal{S}'| = n_{\mathcal{S}}^* - 1 < |\mathcal{S}^*|$, making $\mathcal{S}^*$ non optimal, which is a contradiction. □

Theorem 1. *$\mathcal{R}^*, \mathcal{S}^*$ is a solution to* (7) *if and only if $\mathcal{R}^*$ is a solution to* (8) *and $\mathcal{S}^*$ is a solution to* (9)*.*

Proof. Recall, by construction, rank $C(\mathcal{S}) = |\mathcal{S}|$ for any $\mathcal{S} \subseteq \mathcal{E}$.

First, we prove implication. Assume that $\mathcal{R}^*, \mathcal{S}^*$ is a solution to (7). From Lemma 2 and the constraints of (7), it is evident that $\mathcal{S}^*$ is a solution to (9). Assume there exists $\mathcal{R}'$, $|\mathcal{R}'| = n_{\mathcal{R}}$ such that rank $L(\mathcal{R}')$ > rank $L(\mathcal{R}^*)$. Because of Lemma 1, we can find $\mathcal{S}'$ satisfying (9). It is easy to check that the pair $\mathcal{R}', \mathcal{S}'$ lie in the feasible region of (7), and that $|\mathcal{S}'| < |\mathcal{S}^*|$. This implies that $\mathcal{S}^*$ is not an optimal solution. This is a contradiction, so this $\mathcal{R}'$ cannot exist and therefore, rank $L(\mathcal{R}^*) \geq$ rank $L(\mathcal{R})$ for any $\mathcal{R}$, $|\mathcal{R}| = n_{\mathcal{R}}$, so $\mathcal{R}^*$ is a solution to (8).

Now we proceed to necessity. Assume that $\mathcal{R}^*$ is a solution to (8) and $\mathcal{S}^*$ is a solution to (9), so $|\mathcal{S}^*| = n_{\mathcal{E}} - \text{rank } L(\mathcal{R}^*)$. Consider that the pair $\mathcal{R}', \mathcal{S}'$ is an optimal solution to (7), so $|\mathcal{S}'| \leq |\mathcal{S}^*|$. By construction, it must be that rank $L(\mathcal{R}^*) \geq$ rank $L(\mathcal{R}')$, and by Lemma 2, rank $L(\mathcal{R}^*) \geq n_{\mathcal{E}} - |\mathcal{S}'|$. This implies that $|S^*| \leq |\mathcal{S}'|$, and thus, $|\mathcal{S}^*| = |\mathcal{S}'|$, so the pair $\mathcal{R}^*, \mathcal{S}^*$ is also an optimal solution to (7). □

As these problems can be solved separately, complexity is reduced. However, problem (8) still requires to try out $O((n_{\mathcal{N}}/n_{\mathcal{R}})^{n_{\mathcal{R}}})$ combinations, and problem (9) has $O(n_{\mathcal{E}}^{\text{rank } L(\mathcal{R}^*)})$ possible combinations which in turns require to calculate the rank of big matrices. In the following sections we provide efficient solutions to these problems.

4. Optimal location of turning ratio sensors

Theorem 2 (later in this section) allows to write rank of matrix $L(\mathcal{R})$ for any $\mathcal{R}$ as a function of the number of intersections in the network and their out-degrees, i.e.

$$\text{rank } L(\mathcal{R}) = n_{\mathcal{N}} - n_{\mathcal{R}} + \sum_{k \in \mathcal{R}} \deg^{\text{out}}(k). \tag{10}$$

Using this expression, problem (8) can be rewritten as

$$\mathcal{R}^* = \begin{array}{ll} \underset{\mathcal{R}}{\text{argmax}} & \sum_{k \in \mathcal{R}} \deg^{\text{out}}(k) \\ \text{subject to} & |\mathcal{R}| = n_{\mathcal{R}} \end{array}. \tag{11}$$

To solve this problem, we propose the following algorithm:

Algorithm 1. Location of turning ratio sensors

Inputs: Directed graph $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ and number of turning ratio sensors $n_{\mathcal{R}}$.
Output: Set of intersections $\mathcal{R}^*$.

1. Calculate the vector of out-degrees: $\mathbf{d}(k) \leftarrow \deg^{\text{out}}(k)$, $\forall k \in \mathcal{N}$.
2. Sort $\mathbf{d}$ from highest to lowest and return the sorting vector $\boldsymbol{\lambda}$, i.e. $\boldsymbol{\lambda}(1)$ is the index of the highest element of $\mathbf{d}$, $\boldsymbol{\lambda}(2)$ is the index of the second highest, and so on.
3. Construct $\mathcal{R}^* \leftarrow \{\boldsymbol{\lambda}(1), \boldsymbol{\lambda}(2), \ldots, \boldsymbol{\lambda}(n_{\mathcal{R}})\}$. □

Remark: Several solutions are possible depending on the multiplicity of the out-degrees of the nodes of the network, however, all these solutions are optimal.

Example 2. Consider the traffic network shown in Figure 1. Furthermore, assume that $n_{\mathcal{R}} = 2$. By applying Algorithm 1 to these inputs we obtain:

1. The vector of out-degrees is $\mathbf{d} = [1 \quad 2 \quad 3 \quad 1 \quad 1 \quad 2]^T$.

2. After sorting $\mathbf{d}$, the resulting sorting vector is $\boldsymbol{\lambda} = [3 \quad 2 \quad 6 \quad 1 \quad 4 \quad 5]^T$.
3. The output is found by selecting the first two elements of $\boldsymbol{\lambda}$: $\mathcal{R}^* = \{2, 3\}$.

Note however that there are several nodes with the same out-degree. Because of this, the set $\{3, 6\}$ would also be a solution to problem (11). □

The following propositions show that problem (8) can be converted into problem (11).

Lemma 3. *For any feasible traffic network $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ (see Definition 1) and any set of intersections $\mathcal{R} \subseteq \mathcal{N}$, matrix $A(\mathcal{R})$ is full row rank.*

Proof. Without loss of generality assume that the ordering of the elements of $\mathcal{E}$ is such that the smaller indexes correspond to $\mathcal{E}_{\text{in}}$, followed by $\mathcal{E}_{\text{net}}$ and ending with $\mathcal{E}_{\text{out}}$. Denote $|\mathcal{E}_{\text{in}}| = n_{in}$ and $|\mathcal{E}_{\text{out}}| = n_{out}$.

For now, let $\mathcal{R} = \mathcal{N}$. Using the proposed indexing, we can write

$$A(\mathcal{N})_{i,j} = \begin{cases} 1 & \text{if} \quad i = j + n_{in} \\ -r_{j,i+n_{in}} & \text{else} \end{cases},$$

for $i = 1, 2, \ldots, n_{\mathcal{E}} - n_{in}$ and $j = 1, 2, \ldots, n_{\mathcal{E}}$.

We can split $A(\mathcal{N}) = [X \quad Y]$ where Y is a square matrix of size $n_{\mathcal{E}} - n_{in}$ and can be written as $Y = \mathbb{I} - R^T$ with $R_{i,j} = r_{i+n_{in},j+n_{in}}$ for $i, j \in \{1, 2, \ldots, n_{\mathcal{E}} - n_{in}\}$. Matrix R has the following properties:

1. The diagonal entries of R are zero as $r_{i,i} = 0$.
2. All roads in $\mathcal{E}_{\text{out}}$ have no downstream neighbors, so $r_{i,j} = 0 \ , \ \forall i \in \mathcal{E}_{\text{out}}$. Thus, the last n_{out} rows of R are zero. Nevertheless, by flow conservation, all the other rows of R sum to 1.
3. For every row i of R there is a sequence of nonzero elements of R of the form $R_{i,i_1}, R_{i_1,i_2}, \ldots, R_{i_q,j}$ such that $j \in \mathcal{E}_{\text{out}}$.

Properties 1 and 2 imply that Y^T is weakly diagonally dominant matrix, i.e. $Y_{i,i} \geq |\sum_{j \neq i} Y_{i,j}|$, where the strict inequality is true only for the last n_{out} rows. Nevertheless, property 3 states that for every row i, there is a sequence of nonzero elements that connect row i to one of the last n_{out} rows. Because of this, Y^T is a special type of matrix called *weakly chained diagonally dominant*, which is known to be non-singular [Shivakumar and Chew (1974)]. Thus all rows of $A(\mathcal{N})$ form a linearly independent set.

For any arbitrary $\mathcal{R} \subseteq \mathcal{N}$, the resulting matrix $A(\mathcal{R})$ is just a reduced version of $A(\mathcal{N})$ with some of its rows removed. As the rows $A(\mathcal{N})$ are linearly independent, it is straightforward that $A(\mathcal{R})$ is full row rank as well. □

Lemma 4. *For any feasible traffic network $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ (see Definition 1) and any set of intersections $\mathcal{U} \subseteq \mathcal{N}$, matrix $B(\mathcal{U})$ is full row rank.*

Proof. Initially assume that $\mathcal{U} = \mathcal{N}$. By construction, each column of $B(\mathcal{N}) \in \{-1, 0, 1\}^{n_{\mathcal{N}} \times n_{\mathcal{E}}}$ has only 2 non-zero entries which sum to 0. The only exceptions are columns corresponding to indexes $\mathcal{E}_{\text{in}}$ which have only one non-zero element

equal to -1, and columns corresponding to indexes $\mathcal{E}_{\text{out}}$ with one non-zero entry equal to 1. Note that $B(\mathcal{N})$ is the incidence matrix of $\{\mathcal{N}, \mathcal{E}\}$.

We proceed by contradiction. Assume that $B(\mathcal{N})$ is not full rank. This implies that there exist two sets $\mathcal{U}_1 \subset \mathcal{N}$ and $\mathcal{U}_2 \subset \mathcal{N}$ such that $\sum_{k\in\mathcal{U}_1} B(\mathcal{N})_{k,j} = -\sum_{k\in\mathcal{U}_2} B(\mathcal{N})_{k,j}$, $\forall j$, which is equivalent to $\bigcup_{k\in\mathcal{U}_1} \mathcal{O}(k) = \bigcup_{k\in\mathcal{U}_2} I(k)$ and $\bigcup_{k\in\mathcal{U}_1} \mathcal{I}(k) = \bigcup_{k\in\mathcal{U}_2} O(k)$. This means that for every node in $\mathcal{U}_1$, any downstream neighbor must be a member of $\mathcal{U}_1$ or $\mathcal{U}_2$. Subsequently, any downstream chain of nodes must be contained in one of these two sets. As every edge is part of path ending in an element of $\mathcal{E}_{\text{out}}$, the previous statements imply that there exists a node $k \in \mathcal{N}$ such that $\mathcal{I}(k) \cap \mathcal{E}_{\text{out}} \neq \varnothing$, which is a contradiction. Hence, the rows of $B(\mathcal{N})$ form a linearly independent set.

Following a similar discussion as at the end of the proof of Lemma 3, we note that for $\mathcal{U} \subseteq \mathcal{N}$, the corresponding $B(\mathcal{U})$ implies only the removal of rows from $B(\mathcal{N})$, so the remaining rows are still linearly independent, and thus $B(\mathcal{U})$ is full row rank. □

Theorem 2. *For any feasible network* $\{\mathcal{C}\cup\mathcal{N}, \mathcal{E}\}$*, and sets* $\mathcal{R} \subseteq \mathcal{N}$ *and* $\mathcal{U} = \mathcal{N}\backslash\mathcal{R}$*, it holds that rank* $L(\mathcal{R})$ = *rank* $A(\mathcal{R})$ + *rank* $B(\mathcal{U})$*.*

Proof. For an arbitrary $k \in \mathcal{N}$, we can check that $\mathbb{1}^T A(\{k\}) = B(\{k\})$. This is due to the fact that $\sum_i r_{i,j} = 1$ for all j, and that $r_{i,j} = 0$ if j is not a downstream neighbor of i. Conversely, $\mathbb{1}^T A(\{k\})$ and $B(\{p\})$ share no non-zero entries for $k \neq p$, because $\mathcal{O}(k) \cap \mathcal{O}(p) = \varnothing$ and $\mathcal{I}(k) \cap \mathcal{I}(p) = \varnothing$ if $k \neq p$. Thus, for an arbitrary set $\mathcal{K} \subseteq \mathcal{N}$, each row of $B(\mathcal{K})$ can be obtained by the combination of unique rows of $A(\mathcal{K})$. As $\mathcal{R} \cap \mathcal{U} = \varnothing$, no row of $B(\mathcal{U})$ can be written as a combination of rows of $A(\mathcal{R})$, so both matrices have linearly independent rows. □

5. Optimal location of flow sensors

In this section we discuss an efficient solution to problem (9). The following Corollary provides a way to calculate the optimal number of flow sensors n_s^*.

Corollary 1. *For any feasible traffic network* $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ *and any set of intersections* $\mathcal{R}^* \subseteq \mathcal{N}$ *with cardinality* $n_{\mathcal{R}}$*, the minimum number of flow sensors required to infer all flows in the network is*

$$n_s^* = n_{\mathcal{E}} - n_{\mathcal{N}} + n_{\mathcal{R}} - \sum_{k\in\mathcal{R}^*} deg^{out}(k). \tag{12}$$

Proof. Let $\mathcal{S}^*$ be an optimal solution to (9), so $n_{\mathcal{S}}^* = |\mathcal{S}^*|$ = rank $C(\mathcal{S}^*)$ = $n_{\mathcal{E}}$ − rank $L(\mathcal{R}^*)$. Thus, the corollary follows directly from Theorem 2. □

To solve the sensor location problem we require a method that locates the number of sensors indicated in the corollary in such a way that the rows of $\mathcal{C}(\mathcal{S}^*)$ and the rows of $L(\mathcal{R}^*)$ are linearly independent.

Because of the efficiency and simplicity of graph-based approaches such as the one of He (2013), we decided to expand these techniques to include partial

information of turning ratios. We propose Algorithm 2 which makes use of the topological structure of the traffic network and the information given by the set $\mathcal{R}^*$. The algorithm creates spanning trees of the input graph by using the well known Depth First Search (DFS), which is a well known graph traversal algorithm.

Algorithm 2. Location of flow sensors

Inputs: Directed graph $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ and set $\mathcal{R}^* \subseteq \mathcal{N}$.
Output: Set of measured edges $\mathcal{S}^*$.

1. Replace all nodes in $\mathcal{C}$ with a single node v_0 such that $\mathcal{E}_{\text{in}} = \mathcal{O}(v_0)$ and $\mathcal{E}_{\text{out}} = \mathcal{I}(v_0)$.
2. Initialize $\mathcal{E}_R \leftarrow \varnothing$ and $\mathcal{E}' \leftarrow \mathcal{E}$. For each $k \in \mathcal{R}^*$:
   2.1 Find $\{e_1, e_2, \ldots, e_q\} = \mathcal{O}(k)$.
   2.2 Assign $\mathcal{E}_R \leftarrow \{e_2, \ldots, e_q\}$.
   2.3 Assign $\mathcal{E}' \leftarrow \mathcal{E}' \setminus \mathcal{E}_R$.
3. Ignoring edge direction, perform a depth first search (DFS) over $\{\mathcal{N} \cup \{v_0\}, \mathcal{E}'\}$ starting at v_0. Denote $\mathcal{N}_T$ and $\mathcal{E}_T$ as the visited nodes and edges, respectively.
4. While $\mathcal{R}^* \setminus \mathcal{N}_T \neq \varnothing$:
   4.1 For each $k \in \mathcal{R}^* \setminus \mathcal{N}_T$:
      4.1.1 Construct $\mathcal{M}_k = \{m \in \mathcal{N} \mid \exists j \in \mathcal{O}(k) \cap \mathcal{I}(m) \cap \mathcal{E}_R\}$.
      4.1.2 If $\exists m \in \mathcal{M}_k \cap \mathcal{N}_T$:
         4.1.2.1 Find $j \in \mathcal{O}(k) \cap \mathcal{I}(m)$.
         4.1.2.2 Return k and j. Exit loop.
   4.2 Find $e_1 \in \mathcal{O}(k) \cap \mathcal{E}'$.
   4.3 Assign $\mathcal{E}' \leftarrow \mathcal{E}' \setminus \{e_1\}$
   4.4 Assign $\mathcal{E}_R \leftarrow (\mathcal{E}_R \setminus \{j\}) \cup \{e_1\}$
   4.5 Ignoring edge direction, perform a DFS over $\{\mathcal{N} \cup v_0, \mathcal{E}'\}$ starting at k. Denote $\mathcal{N}_k$ and $\mathcal{E}_k$ the visited nodes and edges, respectively.
   4.6 Assign $\mathcal{N}_T \leftarrow \mathcal{N}_T \cup \mathcal{N}_k$.
   4.7 Assign $\mathcal{E}_T \leftarrow \mathcal{E}_T \cup \mathcal{E}_k \cup \{j\}$.
5. Assign $\mathcal{S}^* = \mathcal{E}' \setminus \mathcal{E}_T$.

Remark: for the case $\mathcal{R}^* = \varnothing$, only steps 1, 3 and 5 are performed, and our algorithm becomes the same as the one presented in He (2013).

The first step of the algorithm aggregates the sources and sinks into a single node. The resulting node will satisfy the flow conservation equations and will make the graph strongly connected. Step 2 removes all but one of the outgoing edges for each of the nodes in $\mathcal{R}^*$. Then, step 3 constructs a tree from the remaining edges. However, because of the removal of edges in step 2, the resulting graph may become disconnected, so the DFS algorithm may not reach all of the nodes in the graph. Step 4 redoes the removal of outgoing edges from $\mathcal{R}^*$ such that the resulting graph is connected, and finishes the construction of

a spanning tree of the original graph. Finally, flow sensors are located in edges which are not included in the spanning tree or the removed edges.

Remark: The removal of links in step 2 and the construction of spanning trees using the DFS algorithm are not unique, and alternative node and edge indexing might yield different sensor configuration. Nevertheless, all multiple solutions provide the same number of sensors, and are thus equally optimal.

Example 3. Consider the traffic network shown in Figure 1. Let $\mathcal{R}^* = \{2, 3\}$ as obtained in Example 2. The optimal location of flow sensors is given by Algorithm 2: the application of step 1 will generate the graph shown in Figure 2.

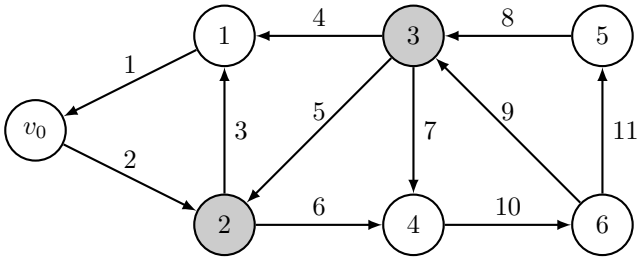

Figure 2: Graph after Step 1.

During step 2 the algorithm iterates over the elements of $\mathcal{R}^*$. For intersection 2, it can be seen that $\mathcal{O}(2) = \{3, 6\}$, where edge 6 is arbitrarily selected and added to set $\mathcal{E}_R$. Similarly for node 3, $\mathcal{O}(3) = \{4, 5, 7\}$, and edges 4 and 5 are arbitrarily selected and added to $\mathcal{E}_R$. Then, $\mathcal{E}_R = \{4, 5, 6\}$ is removed from the graph. The resulting graph is shown in Figure 3.

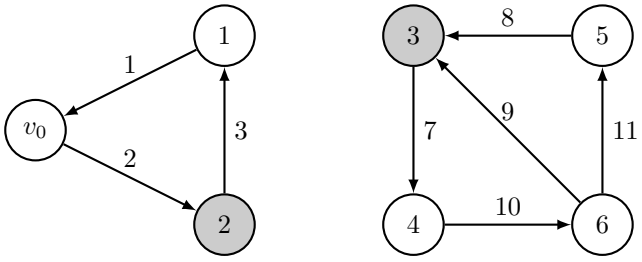

Figure 3: Graph after Step 2.

Next, step 3 performs a DFS staring at v_0 yielding $\mathcal{N}_T = \{v_0, 1, 2\}$ and $\mathcal{E}_T = \{2, 3\}$. However, it can be seen that the graph of Figure 3 is disconnected and that intersection 3 does not belong to the visited nodes $\mathcal{N}_T$. Thus, we proceed with step 4, where we search for a node in $\mathcal{R}^*$ that has an outgoing edge that connects to one of the nodes in $\mathcal{N}_T$, i.e. node 3. Recall that in step 2, edge 7 was arbitrarily selected to remain in the graph. We now remove edge 7

from the graph and add it to $\mathcal{E}_R$. Then, we perform a DFS staring from node 3, adding the visited nodes $\mathcal{N}_3 = \{3, 4, 5, 6\}$ and edges $\mathcal{E}_3 = \{8, 10, 11\}$ to $\mathcal{N}_T$ and $\mathcal{E}_T$. Finally, edge 4 is removed from $\mathcal{E}_R$ and added to $\mathcal{E}_T$, hence reconnecting the disconnected components (see Figure 4).

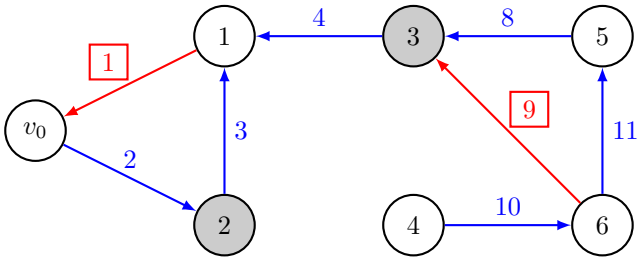

Figure 4: Graph after Step 4.

After these steps, the visited nodes are $\mathcal{N}_T = \{v_0, 1, 2, 3, 4, 5, 6\}$ and the visited edges are $\mathcal{E}_T = \{2, 3, 4, 8, 10, 11\}$. Figure 4 shows the resulting graph, where the edges belonging to $\mathcal{E}_T$ are shown in blue and the remaining edges are marked in red. Note that the subgraph $\{\mathcal{N}_T, \mathcal{E}_T\}$ contains all the nodes, is connected, and has no cycles. Thus, it is a spanning tree of the original input. Finally, sensors are located in the remaining edges $\mathcal{S}^* = \{1, 9\}$, shown with boxes in the figure. □

In Appendix A we show that the output of Algorithm 2 is indeed a solution to Problem (9).

6. Cost minimization

Until now, we considered the case where the number of turning ratio sensors was fixed and the number of flow sensors was minimized. Nevertheless, another case of interest might be when the quantities of both sensors are to be decided such that another variable (e.g. installation cost) is the decision criteria. This can be formulated as,

$$\begin{aligned} \underset{\mathcal{R},\mathcal{S}}{\text{argmin}} \quad & c_{\mathcal{S}}|\mathcal{S}| + c_{\mathcal{R}}|\mathcal{R}| \\ \text{subject to} \quad & \operatorname{rank}\left(\begin{bmatrix} L(\mathcal{R}) \\ C(\mathcal{S}) \end{bmatrix}\right) = n_{\mathcal{E}} \end{aligned}, \tag{13}$$

where $c_{\mathcal{R}}, c_{\mathcal{S}}$ are the overall costs of the turning ratio and flow sensors, respectively. For the rest of this section, we will write $|\mathcal{S}| = n_{\mathcal{S}}$ and $|\mathcal{R}| = n_{\mathcal{R}}$ to simplify notation.

In the previous sections it was shown that for any fixed $n_{\mathcal{R}}$ the minimum number of flow sensors corresponds to

$$n_{\mathcal{S}} = n_{\mathcal{E}} - n_{\mathcal{N}} + n_{\mathcal{R}} - \sum_{k \in \mathcal{R}} \deg^{\text{out}}(k).$$

Without loss of generality, assume a permutation of the node indexing such that $\deg^{\text{out}}(k) \geq \deg^{\text{out}}(k+1)$ for all $k \in \mathcal{N}$. This can be achieved by the sorting operation used in Algorithm 1. With this indexing, we define

$$f(k) = \sum_{m=1}^{k} \deg^{\text{out}}(m) \quad , \quad k \in \mathcal{N}. \tag{14}$$

To solve (13), first find the value of $n_{\mathcal{R}}$ that satisfies

$$\begin{array}{ll} \underset{n_{\mathcal{R}}}{\text{argmin}} & c_{\mathcal{S}} n_{\mathcal{S}} + c_{\mathcal{R}} n_{\mathcal{R}} \\ \text{subject to} & n_{\mathcal{S}} = n_{\mathcal{E}} - n_{\mathcal{N}} + n_{\mathcal{R}} - f(n_{\mathcal{R}}) \\ & 0 \leq n_{\mathcal{R}} \leq n_{\mathcal{N}} \end{array} , \tag{15}$$

and then use this number as an input to Algorithms 1 and 2. It is straightforward to check that this procedure provides a solution to (13): for any $n_{\mathcal{R}}$, it was shown that the algorithms generate sets $\mathcal{S}$ and $\mathcal{R}$ that satisfies the constraints of (13) and (15).

Problem (15) can be simplified to

$$\begin{array}{ll} \underset{n_{\mathcal{R}}}{\text{argmin}} & (c_{\mathcal{S}} + c_{\mathcal{R}}) n_{\mathcal{R}} - c_{\mathcal{S}} f(n_{\mathcal{R}}) \\ \text{subject to} & 0 \leq n_{\mathcal{R}} \leq n_{\mathcal{N}} \end{array} . \tag{16}$$

Due to the choice of node indexing, $\deg^{\text{out}}(k)$ is non-increasing, and therefore, $f(k)$ is concave. The cost function in (16) is then convex, so gradient-based approaches are guaranteed to work. After straightforward manipulations we find that the optimal $n_{\mathcal{R}}$ is such that

$$\deg^{\text{out}}(n_{\mathcal{R}}) \geq \frac{c_{\mathcal{S}} + c_{\mathcal{R}}}{c_{\mathcal{S}}} \geq \deg^{\text{out}}(n_{\mathcal{R}} + 1). \tag{17}$$

This problem is easily solved by performing a simple search and returning the corresponding index. To satisfy the constraints, if $(c_{\mathcal{S}} + c_{\mathcal{R}})/c_{\mathcal{S}}$ is greater than $\deg^{\text{out}}(1)$, then $n_{\mathcal{R}} = 0$. On the other hand, if this ratio is 1, we let $n_{\mathcal{R}}$ to be the largest integer for which $\deg^{\text{out}}(n_{\mathcal{R}}) = 2$. This is because locating turning ratio sensors in nodes with only one outbound edge will not provide any useful information.

Note that the case where $(c_{\mathcal{S}} + c_{\mathcal{R}})/c_{\mathcal{S}}$ is an integer, there may be multiple equally optimal solutions to the problem.

7. Computational complexity

7.1. Sensor location

To solve the original problem postulated in (7), it was shown that it could be split into two simpler problems (8) and (9), for which the solutions are given using Algorithms 1 and 2 respectively. Therefore, the total computational

complexity of the proposed method is the sum of the complexities of these two algorithms.

For Algorithm 1, the only required operation is the sorting of the vector of out-degrees, which is known to have complexity $O(n_{\mathcal{N}} \log n_{\mathcal{N}})$. For Algorithm 2, it can be seen that steps 3 and 4 contain the more complex operations, as they require to perform graph traversing and to search for particular structures. It was discussed that step 2 of Algorithm 2 could generate a disconnected graph with q connected components. As each of the subgraphs must contains at least one node with an outgoing edge the previously connected to another subgraph, it must be that $q \leq n_{\mathcal{R}}$. For each of the subgraphs $\{\mathcal{N}_i, \mathcal{E}_i\}$, $i = 0, \ldots, q-1$, a Depth First Search is performed (Steps 3 and 4). As each of this searches has complexity $O(|\mathcal{N}_i| + |\mathcal{E}_i|)$, the total complexity is

$$\sum_{i=0}^{q-1} O\left(|\mathcal{N}_i| + |\mathcal{E}_i|\right) = O\left(\sum_{i=0}^{q-1} |\mathcal{N}_i| + \sum_{i=0}^{q-1} |\mathcal{E}_i|\right) = O(n_{\mathcal{N}} + n_{\mathcal{E}}).$$

On the other hand, step 4 requires to find for every subgraph an edge that connects to the main graph. This search operation has an average complexity of $O(\log n_{\mathcal{N}})$ that must be repeated at most $n_{\mathcal{R}}$ times.

By taking into account all the operations from both algorithms, we obtain a total complexity of

$$O(n_{\mathcal{N}} \log n_{\mathcal{N}} + n_{\mathcal{E}}).$$

This represents a significant improvement when compared to other works in the literature such as Hu et al. (2009) and Ng (2012) that rely on Gaussian elimination, and have computational complexity of $O(n_{\mathcal{N}} n_{\mathcal{E}}^2)$.

7.2. Flow reconstruction

To calculate the flow value for all roads in the network, we use the inversion of system (5). It has been shown that the result of Algorithms 1 and 2 generates a square matrix $\begin{bmatrix} L(\mathcal{R}^*) \\ C(\mathcal{S}^*) \end{bmatrix}$ with complete rank. Without loss of generality, assume an indexing of roads such that $C(\mathcal{S}^*) = [\mathbf{0} \quad \mathbb{I}]$. The flow vector $\boldsymbol{\varphi}$ can be split into two components, namely the measured flows $\boldsymbol{\varphi}_m$ and the unmeasured flows $\boldsymbol{\varphi}_u$, such that $\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{\varphi}_u \\ \boldsymbol{\varphi}_m \end{bmatrix}$. Additionally, consider a partition $L(\mathcal{R}^*) = [L_u \quad L_m]$, where L_u is a square matrix. Because the matrix $\begin{bmatrix} L_u & L_m \\ \mathbf{0} & \mathbb{I} \end{bmatrix}$ is full rank, it is easy to show that L_u is invertible. After straightforward calculations, it can be seen that

$$\boldsymbol{\varphi}_u = -L_u^{-1} L_m \boldsymbol{\varphi}_m. \tag{18}$$

As the values of φ_m are known, the whole flow vector can be reconstructed. This calculation has a computational complexity $O(n_{\mathcal{N}}^3 + n_{\mathcal{R}}^3 - n_{\mathcal{S}}^3)$, which is a reduction with respect to the original matrix inversion.

These results are comparable with other works in the literature, which also involve matrix inversion. However, it should be noted that the algorithm developed by He (2013) was able to reconstruct road flows in quasi-linear complexity, albeit only for the case $\mathcal{R} = \varnothing$. It is unclear if this method can be extended to the general case $\mathcal{R} \subseteq \mathcal{N}$ in order to reduce complexity. These extensions are to be analyzed in future research.

8. Study case 1: grid-like network

In this section, we evaluate the performance of the proposed method by using a randomly generated network, which is shown in Figure 5. This network has a Manhattan-like structure with randomly added and deleted edges to better represent a real urban traffic grid. The network contains 24 source/sink nodes, 117 intersections and 254 roads. For evaluation purposes, we consider two different experiments. In the first one, we analyze the trade-off between the two different sensing technologies. In the second, we evaluate the performance of flow reconstruction, even when the underlying assumption of steady-state condition is not satisfied by using a dynamic traffic simulation.

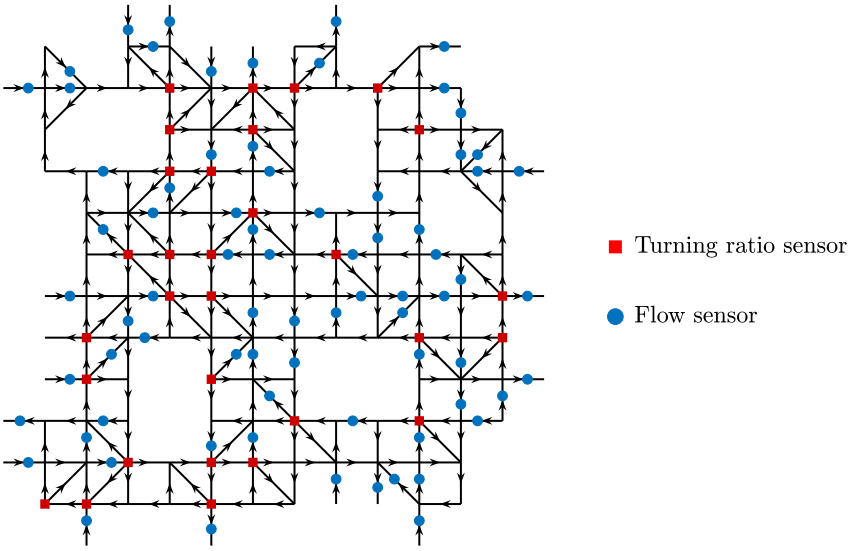

Figure 5: Simulated network with 24 source/sink nodes, 117 intersections and 254 edges. Optimal locations of turning ratio and flow sensors are shown for the case $n_{\mathcal{R}} = 30$ and $n_{\mathcal{S}}^* = 75$.

8.1. Location of sensors

For a given value of $n_{\mathcal{R}}$ we solved problems (9) and (11). The process was done iteratively by using different values of $n_{\mathcal{R}}$ varying from 0 to $|\mathcal{N}|$. In Figure 6

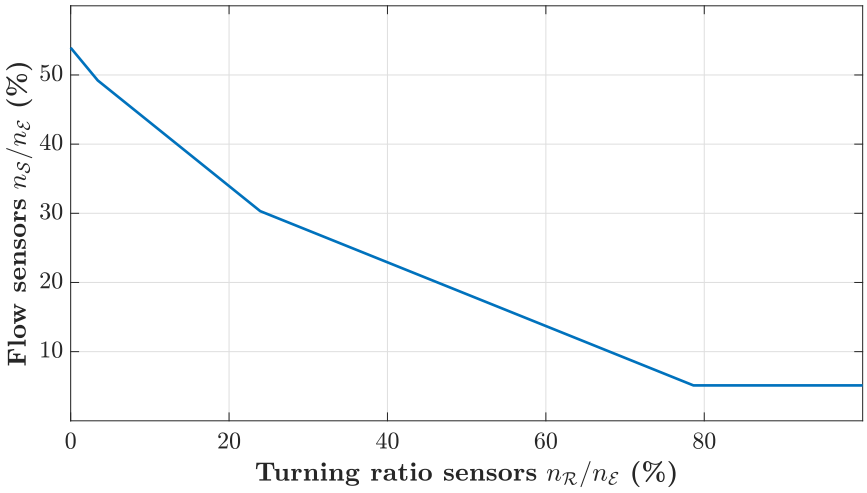

Figure 6: Trade off between turning ratio and flow sensors for the simulated network. The x-axis represents the ratio of monitored intersections $n_\mathcal{R}/n_\mathcal{N}$ and the y-axis is the ratio between flow sensors and roads $n_\mathcal{S}/n_\mathcal{E}$.

we show the required number of flow sensors for each value of $n_\mathcal{R}$. It can be seen that when $n_\mathcal{R} = 0$, it is required to monitor more than half of the roads, result which is coherent with those shown by Ng (2012) and Viti et al. (2014), who estimated that for real networks it is required to monitor up to 60% of all roads. By including the information of turning ratios, we can effectively reduce the number of flow sensors. The different slopes of the plot correspond to the distribution of the out degrees of the nodes in the network. Note that when nodes have several outgoing edges, the rate of decrease of sensors is highest. As expected, when intersections have only one outgoing edge, placing turning ratio sensors give no information at all and the number of sensors stays constant.

As an example, Figure 5 shows the sensor configuration obtained by arbitrarily setting $n_\mathcal{R} = 30$ and running the algorithms. With this inputs, 75 flow sensors were required. For comparison purposes, we also implemented the Gaussian reduction method used in Hu et al. (2009) and Ng (2012), which yielded the same number of sensors. The running time of both methods was also measured: our graph based approach took 0.24 seconds in average, whereas the algebraic approach took approximately 0.19 seconds. As the network size is small, the measured times are similar.

8.2. Dynamic traffic simulation

To evaluate the performance of the flow reconstruction method, we used this network to perform a dynamic simulation of traffic flows. We use the Cell Transmission Model (CTM) which is a discretization of the well known LWR traffic model [Daganzo (1995)]. In the CTM, each road $i \in \mathcal{E}$ has associated three variables: upstream flow $\varphi_i^{up}(t)$, downstream flow $\varphi_i^{dn}(t)$, and vehicular

Table 2: Parameters for the dynamic traffic simulation

Parameter	**Symbol**	**Value**
Road length	L	150 m
Time step	Δt	0.5 s
Free flow speed	v_0	60 km/h
Jam density	ρ^{max}	125 veh/km
Capacity	φ^{max}	2400 veh/h
Critical density	ρ_c	40 veh/km
Congested wave propagation speed	ω	28.2 km/h

density $\rho_i(t)$. The equation of the CTM is given as

$$\rho_i(t+\Delta t) = \rho_i(t) + \frac{\Delta t}{L_i}\left[\varphi_i^{up}(t) - \varphi_i^{dn}(t)\right], \tag{19}$$

where Δt is the discretized time step and L_i is the road length. Furthermore, at each iteration the upstream and downstream flows are must be specified. To do this, the CTM uses the triangular fundamental diagram coupled with a junction model,

$$\begin{aligned} \boldsymbol{\varphi}^{dn}(t) \quad = \quad & \underset{\boldsymbol{\varphi}^{dn}(t)}{\text{argmax}} \quad \sum_{i=1}^{n_\varepsilon} \varphi_i^{dn}(t) \\ & \text{subject to} \quad \varphi_i^{dn}(t) \leq \min\{v_0\rho_i(t), \varphi^{\max}\} \\ & \qquad\qquad \sum_{i=1}^{n_\varepsilon} r_{i,j}\varphi_i^{dn}(t) \leq \min\{\omega(\rho^{\max} - \rho_j(t)), \varphi^{\max}\} \\ & \qquad\qquad 0 \leq \varphi_i^{dn}(t) \leq \varphi^{\max} \end{aligned},$$

where the free flow speed v_0, the capacity φ^{max}, the jam density ρ^{max}, and the congested wave propagation speed ω, are the parameters of the triangular fundamental diagram. The critical density $\rho_c = \varphi^{max}/v_0$ is defined as the density when traffic flow is at its maximum; all roads with density below ρ_c are said to be in free flow, whereas roads with density above ρ_c are said to be in congestion. For the simulations we used the values shown in Table 2[1].

The turning ratios used in the simulation where chosen to reflect that most of the congestion is only present in a few main roads, whereas most of the secondary roads are lightly used. The values were calculated in the following manner: initially, all of the possible turns were given the same priority by assigning each one a constant weight. Next, for each source/sink node pair the shortest path

[1]Because of numerical stability, the parameter Δt must be chosen such that $L/\Delta t > v_0$

was identified. The turns that were contained in these shortest paths were given a higher priority by increasing their values with a predefined amount. Finally, the ratios where normalized such that the condition $\sum_j r_{i,j} = 1$ was satisfied. Because of this choice, vehicular flow is highest by following the shortest paths from the entries of the network towards the exits, but a small percentage takes other directions. Thus, the flow and density distribution of the simulation are non-homogeneous among the roads.

Remark: The selection of turning ratios is a simplification of real world scenarios. For a more realistic modeling, *dynamic traffic assignment* models should be employed, which determine the routing behavior by taking into account the network state. This choice was made to simplify the simulation.

The methodology described in this article for sensor location and flow reconstruction is based on a steady state network. This can be stated in the condition $\boldsymbol{\varphi}^{up}(t) = \boldsymbol{\varphi}^{dn}(t) = \boldsymbol{\varphi}$ for all t, which in turn implies that there is no evolution in time of the density $\boldsymbol{\rho}(t + \Delta t) = \boldsymbol{\rho}(t)$. If this condition is not met, then it is expected to obtain errors in the reconstruction of flows. To evaluate the effect of moderate time-varying network conditions, we performed the following experiment:

1. A step input of constant boundary inflows was applied to the network with zero internal flows and zero density as initial conditions.
2. After a period of time, the network reached an equilibrium state. This state was recorded as $\boldsymbol{\varphi}^{ss}$ for flows and $\boldsymbol{\rho}^{ss}$ for the densities.
3. Using the previous steady-state as an initial condition, a second simulation was performed by adding a sine wave to the boundary inflows: $\varphi_i^{up}(t) = \varphi_i^{ss} + A\sin(2\pi t/T)$ for $i \in \mathcal{E}_{\text{in}}$. The period was chosen to be $T = 2$ hours.
4. At every time instant, flow measurements are obtained $\boldsymbol{\varphi}_m(t) = C(\mathcal{S}^*)\boldsymbol{\varphi}^{dn}(t)$.
5. Flow estimates are calculated using (18).
6. The error is quantified using the Normalized Root Mean Squared Deviation (NRMSD):

$$NRMSD(t) = \frac{\sqrt{\dfrac{1}{n_{\mathcal{E}}}\displaystyle\sum_{i=1}^{n_{\mathcal{E}}}[\varphi_i^{dn}(t) - \hat{\varphi}_i(t)]^2}}{\dfrac{1}{n_{\mathcal{E}}}\displaystyle\sum_{i=1}^{n_{\mathcal{E}}}\varphi_i^{dn}(t)}.$$

This process was performed for two different cases, one where all of the roads of the network remain in free flow at every time instant, and another where some of the roads are in congestion.

8.2.1. Free flow case

In the first simulation case, the network was initialized using a constant demand flow of 25% of the capacity, i.e. 600 veh/h. Figure 7 shows the network's

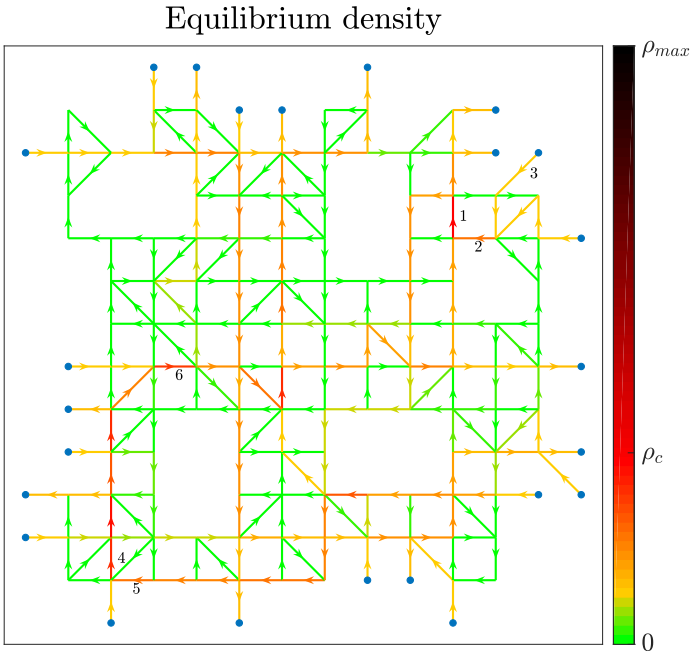

Figure 7: Equilibrium density obtained with a step input of 25% of the maximum flow. Blue points denote source/sink nodes. All roads are in free flow.

equilibrium density, which was obtained 5 minutes after the step impulse was applied. Because of the choice of turning ratios, most of the roads present very light traffic conditions, whereas the shortest paths connecting source and sink nodes (blue dots) present higher densities. Nevertheless, all the roads have densities lesser than the critical value, so the network is in free flow. Using this equilibrium state as an initial condition for a subsequent simulation, the inputs were added a sine wave with an amplitude of 60 veh/h and a period of 2 hours during a total time of 4 hours. Figure 8 shows the network's response to the time-varying input. To simplify the display, only the indexed roads 1-6 in Figure 7 are shown.

From Figure 8 it can be seen that the internal states of the network have a sinusoidal behavior. This is due to two reasons: first, the settling time of the system is very small compared to the wave's period, which causes a quick response of the states to follow the inputs. Second, as the traffic network is in free flow, it can be modeled as a linear system, which explains the propagation of the values of the inputs, albeit in a delayed manner.

To test the reconstruction method, sensors where located using the same configuration as Figure 5, using 30 turning ratio and 75 flow sensors. The NRMSD of the flow reconstruction is shown in Figure 9. As the simulation uses an equilibrium sate as an initial condition, the error at time $t = 0$ is zero. As time progresses, the error shows a time-varying behavior related to

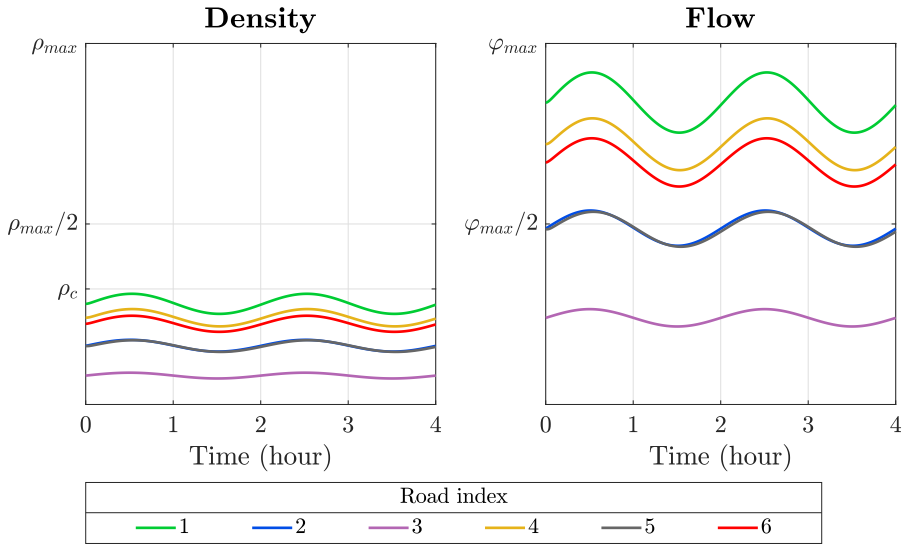

Figure 8: Response of the system to a time varying input. Despite the periods of higher inflows, all the roads remain in free flow. To simplify visualization, only a selection of roads are shown.

the rate of change of the network's state. When the slope of the input signal is zero, $\boldsymbol{\rho}(t+1) \approx \boldsymbol{\rho}(t)$ and the steady-state condition is met, and hence the reconstruction has zero error. However, as the time variation of the system increases, the reconstruction error becomes higher. Nevertheless, even when the basis assumption of our method is not satisfied, we were able to obtain errors under 18%.

8.2.2. Congested case

In an analogous manner to the previous case, we initialized the simulation with zero density for all roads in the network. To obtain a congested situation, the boundary input demands were set to 29.8% of the capacity, i.e 715 veh/h. The network's equilibrium state is shown in Figure 10.

From the figure, it is clear that some of the roads have densities higher than ρ_c, so their equilibrium state is in congestion. This causes the settling time to be higher compared to the free flow case, to just under 20 minutes. This is because many of the origin-destination paths use the same roads, generating bottlenecks and vehicle accumulation in some roads. For example, road 1 has density close to the critical value, which limits the number of vehicles from road 2 that want to enter. The congestion in road 2 travels upstream towards the source node (road 3). This situation is similar to the bottleneck in road 4 that limits the maximum flow in road 5. During our testing, we found that using higher input demands would cause the system to develop gridlock conditions.

Using the equilibrium state as an initial condition, the inputs demands were

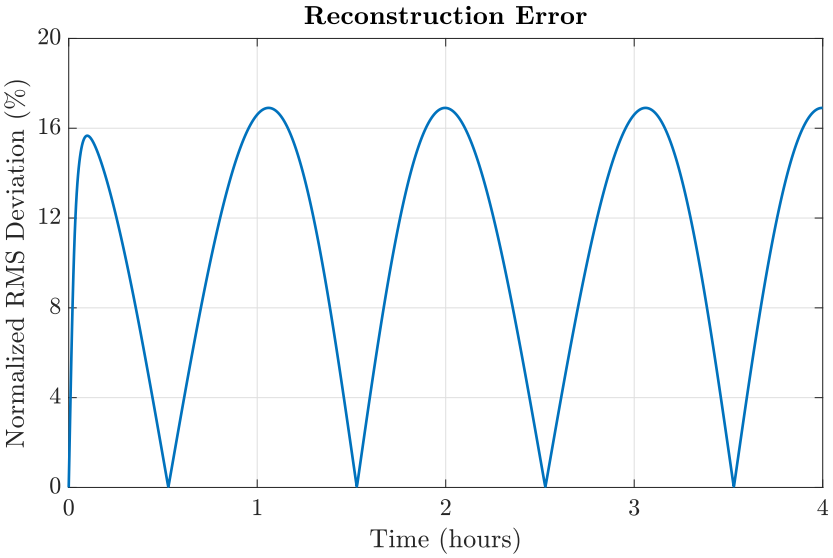

Figure 9: Reconstruction error for the time-varying input simulation.

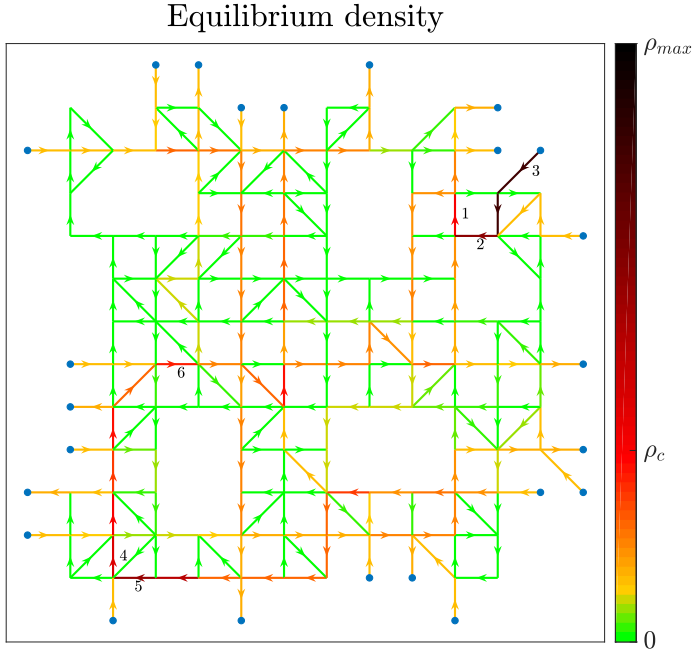

Figure 10: Response of the system to a step input of 29.8% of the maximum flow. The system stabilizes after 30 minutes, with some of the roads in congestion.

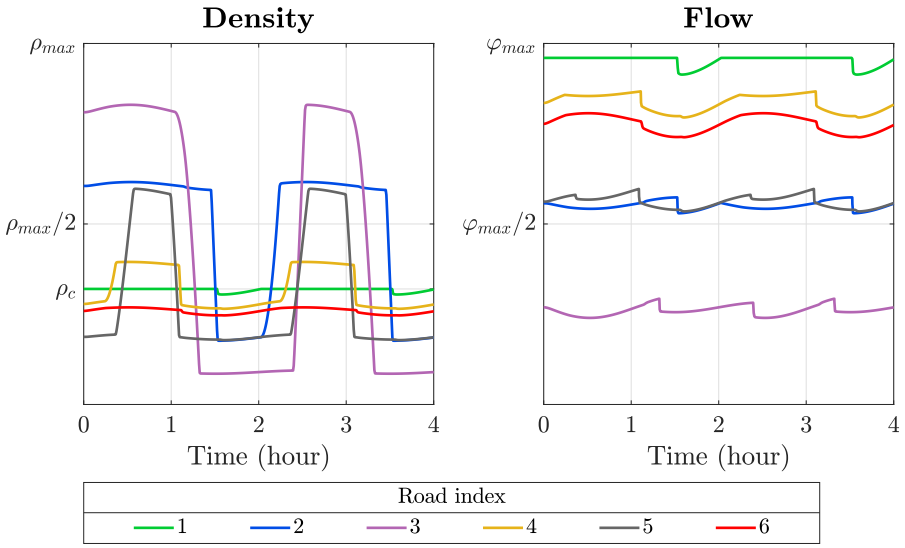

Figure 11: Response of the system to a time varying input. To simplify visualization, only a selection of roads are shown.

added a sine wave with a period of 2 hours and an amplitude of 36 veh/h. The resulting system response is shown in Figure 11, where only the indexed roads 1 to 6 are shown. The non-linearities of the traffic network are evident: some of the roads present saturation, followed by a rapid transition from congestion to free flow when the input demand decreases. This quick transitions cause very high derivatives in the internal states, which drives the system away from the steady-state condition.

The flow reconstruction error is shown in Figure 12. Albeit the non-linearity of the system, the reconstruction error still approaches zero when the time derivative of the system's states is small. Similarly, the error is highest when the rate of change is at its maximum. The rapid increments of the error and the spikes are associated with the quick transitions between congestion and free flow seen in Figure 11.

9. Study case 2: city of Grenoble

The GTL-Ville is an initiative of the Scale-FreeBack[2] project which consists on the analysis of traffic in the city of Grenoble, France. In this context, we are interested in the location of flow and turning ratio sensors which will allow the estimation of the traffic state in the city.

In this paper, we considered a central section of the city spanning an area

[2]Scale-Free control for complex physical network systems, http://scale-freeback.eu.

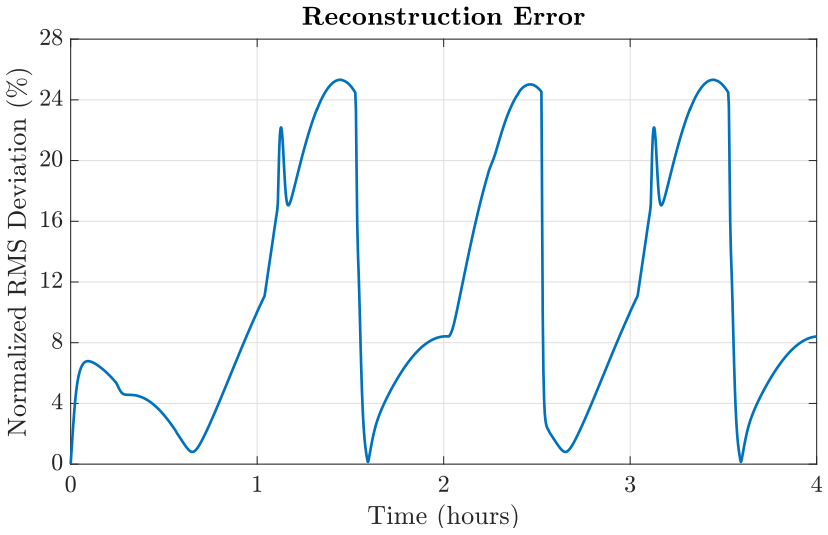

Figure 12: Reconstruction error for the time-varying input simulation.

of approximately 4 km × 4 km. The traffic network is shown in Figure 13. This section was selected because it contains the regularly congested areas of downtown, and some of the principal roads. The nodes that lie outside of the area are considered as source/sink nodes. Additionally, all internal nodes with only outgoing edges or only incoming edges were also labeled in this category, as they do not satisfy the conservation equations. As a result of this preprocessing, the obtained network presents 92 source/sink nodes, 2924 intersections and 5880 roads. It can be seen that the size of the network is considerable, and many of the tools discussed in the literature would fail to solve the sensor location problem in a reasonable amount of time.

To evaluate the algorithms, we proceeded using a similar process as in Section 8: for every value $n_{\mathcal{R}} = 0, 1, \ldots, n_{\mathcal{N}}$, we found the locations of turning ratio and flow sensors using the algorithms. Figure 14, shows the corresponding number of flow sensors for each $n_{\mathcal{R}} \in [0, n_{\mathcal{N}}]$. From Figs. 6 and 14, it can be seen that the knowledge of turning ratios is most useful in the case of complex traffic networks, which can contain intersections with high outdegree, and thus generate regions of very high rate of decrease of the number of sensors. Additionally, for the presented cases, the absolute minimum number of flow sensors that can be obtained (e.g. $\mathcal{R} = \mathcal{N}$) is equal to the number of boundary entering flows $|\mathcal{E}_{\text{in}}|$. This result is in agreement with the work of Lovisari et al. (2016). Figure 14 shows the resulting configuration of turning ratio and flow sensors when $n_{\mathcal{R}} = 1000$ (about 34% of the intersections) and $n_{\mathcal{S}} = 1041$ (about 18% of the roads).

To benchmark the computational complexity of the presented method, the running time of the sensor location algorithms was measured for each iteration. The time values ranged between 5.6 and 7.7 seconds, with a mean of 6.4 sec-

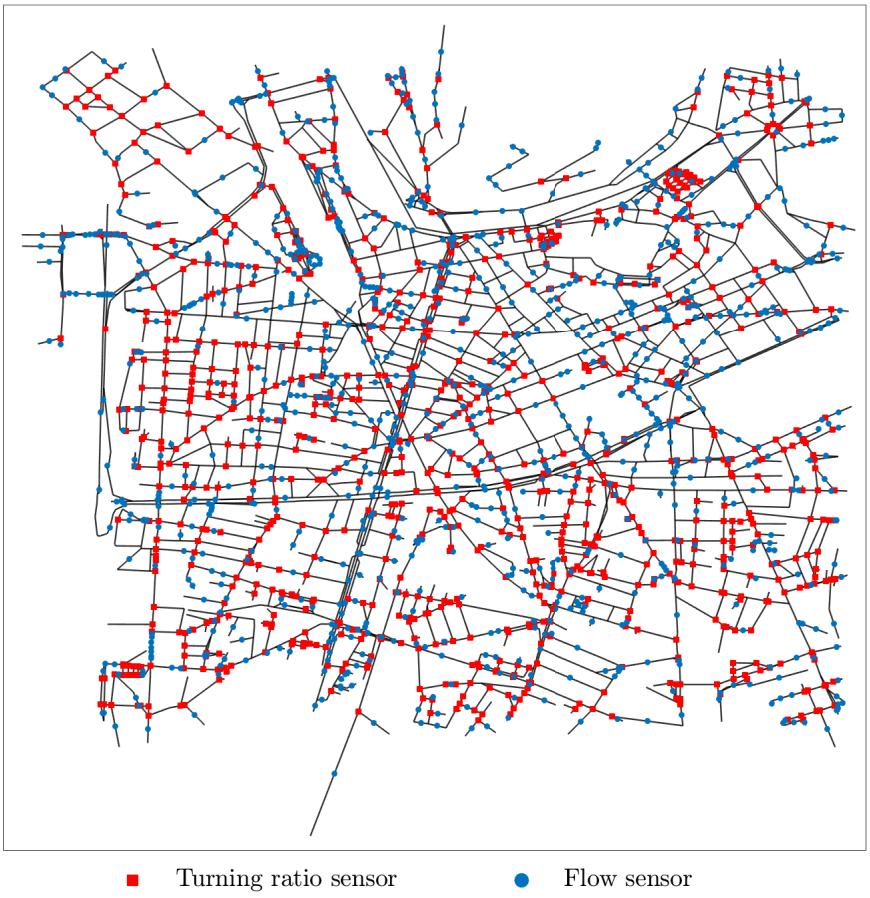

Figure 13: Traffic network of the city of Grenoble, for a section of dimension $\approx$ 4 km $\times$ 4 km. The network has 92 source/sink nodes, 2924 intersections and 5880 roads. The location of flow and turning ratio sensor are presented for the case $n_{\mathcal{R}} = 1000$ and $n_{\mathcal{S}} = 1041$.

onds. For comparison purposes, the method presented in Ng (2012) (which is based on Gaussian reduction) was also implemented for one iteration, yielding a processing time of 32 minutes. This shows that the proposed algorithms are suitable for large traffic networks due to low computational complexity. These times were obtained while using an Intel processor i7-4600U running at 3 GHz clock speed with 8 GB of RAM.

10. Concluding remarks

In this work, we solved the network sensor location problem by using two types of sensing technologies. This allows to reduce the number of flow sensors that are originally required. We solved this problem by finding the solution

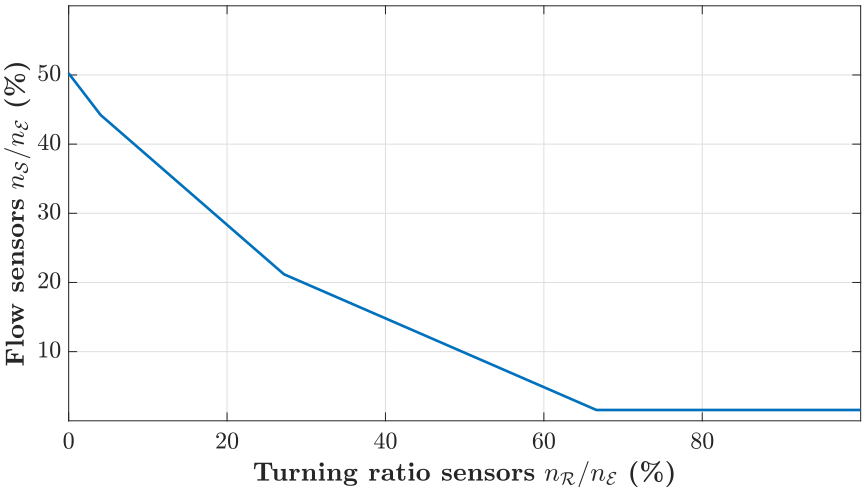

Figure 14: Trade off between turning ratio and flow sensors for the simulated network. The x-axis represents the ratio of monitored intersections $n_\mathcal{R}/n_\mathcal{N}$ and the y-axis is the ratio between flow sensors and roads $n_\mathcal{S}/n_\mathcal{E}$.

of two independent problems: optimally locating a given number of turning ratio sensors, and then, using this result as an input, we locate flow sensors. These problems were shown to have computationally efficient solutions, as the first one only requires vector sorting, and the second one can be solved by the construction of spanning trees. The methods were shown to give an optimal solution to the original problem. The complexity of the proposed algorithms was shown to be $O(n_\mathcal{N} \log n_\mathcal{N} + n_\mathcal{E})$, where $n_\mathcal{N}$ is the number of intersections and $n_\mathcal{E}$ is the number of roads, which is much more efficient than previous methods that rely on algebraic computations providing complexities of $O(n_\mathcal{N} n_\mathcal{E}^2)$.

The proposed methods were tested using two simulated networks. In both cases, it was shown that the knowledge of turning ratios of around 30% on the intersections would require to monitor less than 30% of the roads. Nevertheless, some applications might still find these results to yield unfeasible number of sensors. As future work, we will analyze partial observability to further reduce sensing locations. Additionally, we will investigate on extensions to dynamic traffic networks, in order to be able to reconstruct traffic states in the presence of time-varying input demands.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 694209), http://scale-freeback.eu.

Appendix A. Validity of Algorithm 2

Algorithm 2 presented in Section 5 constructs a basis for the nullspace of $L(\mathcal{R}^*)$ and then locates sensors in such a way that the dimension of $\ker L(\mathcal{R}^*)$ is reduced to 0. Recall that $C(\mathcal{S})$ can be written as the concatenation of rows $\mathbf{u}_s^T$ for $s \in \mathcal{S}$. It can be shown that if there exists any $\mathbf{v} \in \ker L(\mathcal{R}^*)$ such that $\mathbf{v}(i) \neq 0$, then $\mathbf{u}_i^T$ is linearly independent to the rows of $L(\mathcal{R}^*)$, and thus, this row reduces the dimension of the nullspace.

In the following propositions, we show that the proposed algorithm provides a solution to problem (9). First, we show that the algorithm locates exactly the optimal number of flow sensors n_s^*. Subsequently, we show that the collection of $\mathbf{u}_s^T$ for $s \in \mathcal{S}$ is linearly independent to the rows of $L(\mathcal{R}^*)$.

Appendix A.1. Number of sensors

Proposition 2. *Given any feasible network $\{\mathcal{C} \cup \mathcal{N}, \mathcal{E}\}$ and any set $\mathcal{R}^* \subset \mathcal{N}$, the graph $\{\mathcal{N}_T, \mathcal{E}_T\}$ generated by Algorithm 2 is a spanning tree of $\{\mathcal{N} \cup \{v_0\}, \mathcal{E}\}$.*

Proof. From the original graph, we know that every edge is part of a directed path that begins with an element of $\mathcal{E}_{\text{in}}$ and ends with an element of $\mathcal{E}_{\text{out}}$. Because of this, step 1 of the algorithm generates a strongly connected graph: for any node $k \in \mathcal{N}$, there is a directed path from v_0 to k and another from k to v_0.

As step 2 implies the removal of edges, it is possible that the result is a disconnected graph. For now, consider the case when the graph is connected. Step 3 performs a DFS over a connected graph, so every node is visited, and thus step 4 is not performed. Therefore, $\{\mathcal{N}_T, \mathcal{E}_T\}$ is a spanning tree of $\{\mathcal{N} \cup \{v_0\}, \mathcal{E}\}$.

Now consider the case when step 2 generates a disconnected graph. Assume that there are q connected components $\mathcal{G}_i = \{\mathcal{N}_i, \mathcal{E}_i\}$, $i = 0, 1, \ldots, q-1$, and let $\mathcal{G}_0$ be the subgraph containing node v_0. The DFS in step 3 will visit only the nodes in $\mathcal{G}_0$. In step 4, we begin by searching some $k \in \mathcal{R}$ that belongs to some $\mathcal{G}_i$, $i \neq 0$, such that originally there is an edge $j \in \mathcal{O}(k)$ that would connect $\mathcal{G}_i$ and $\mathcal{G}_0$. As the original graph is strongly connected, every node is part of a directed path connecting to $v_0 \in \mathcal{G}_0$, so such k and j must always exist. Next, the single remaining outgoing edge of k is also removed. If this causes $\mathcal{G}_i$ to become disconnected, denote $\mathcal{G}_q$ as the new connected component, which will be treated in the following iterations. The DFS starting from k will generate a spanning tree of $\mathcal{G}_i$. Then, the addition of edge j connects subgraphs $\mathcal{G}_i$ and $\mathcal{G}_0$ without creating any cycles. This process is carried out iteratively until all nodes haven been visited, creating a spanning tree of $\{\mathcal{N} \cup \{v_0\}, \mathcal{E}\}$. □

Corollary 2. *For any set of intersections $\mathcal{R}^*$, with cardinality $|\mathcal{R}^*| = n_{\mathcal{R}}$, the described algorithm locates n_s^* sensors, where n_s^* follows Corollary 1.*

Proof. From step 1, we start with a graph that has $n_{\mathcal{N}} + 1$ nodes and $n_{\mathcal{E}}$ edges. At the end of step 4, the algorithm obtains a spanning tree of this graph, which is known to have $n_{\mathcal{N}}$ edges. It follows that there are $n_{\mathcal{E}} - n_{\mathcal{N}}$ edges not contained in the tree. However, step 2 removes $\deg^{\text{out}}(k) - 1$ edges for each $k \in \mathcal{R}^*$, for a

total of $\sum_{k\in\mathcal{R}^*} \deg^{\text{out}}(k) - n_{\mathcal{N}}$ removed edges. As sensors are to be located in edges not belonging to the tree nor the set of removed edges, we end up with $n_{\mathcal{E}} + n^*_{\mathcal{R}} - \sum_{k\in\mathcal{R}^*} \deg^{\text{out}}(k) - n_{\mathcal{N}}$ locations, which is the same result as Corollary 1. □

Appendix A.2. Linearly independence of sensor locations

The algorithm must also find the location of these sensors such that the rows of $C(\mathcal{S})$ and $L(\mathcal{R}^*)$ are linearly independent. To do this, we first discuss the relationship of the null space of matrix $L(\mathcal{R}^*)$ with the cycles of the graph. Then, we show that by placing sensors in edges that "break" the cycles we obtain a set of linearly independent rows to matrix $L(\mathcal{R}^*)$.

Definition 3. Given a path $\mathcal{P} \subset \mathcal{E}$, $\mathcal{P} = \{e_1, e_2, \ldots, e_l\}$ such that there are no repeated edges, the corresponding path vector is defined as $\mathbf{v} \in \{-1, 0, 1\}^{n_{\mathcal{E}}\times 1}$ where $\mathbf{v}(e_1) = 1$, $\mathbf{v}(e_i) = 0$ if $e_i \notin \mathcal{P}$, and $\mathbf{v}(e_i) = \mathbf{v}(e_{i-1})$ if edges e_i, e_{i-1} have the same direction or $\mathbf{v}(e_i) = -\mathbf{v}(e_{i-1})$ else.

Lemma 5. *Let $\mathcal{P}$ be a cycle that does not include any node belonging to $\mathcal{R}^*$. The corresponding path vector $\mathbf{v}$ belongs to the null space of $L(\mathcal{R}^*)$.*

Proof. If $\mathcal{R}^* = \varnothing$, then $L(\varnothing) = B(\mathcal{N})$ is the incidence matrix of the graph $\{\mathcal{N} \cup \{v_0\}, \mathcal{E}\}$ and the proposition has already been proved by Castillo et al. (2014).

For an arbitrary $\mathcal{R}^*$, as the nodes comprised by $\mathcal{P}$ do not belong to $\mathcal{R}^*$, then each one of them is associated to a row of $B(\mathcal{U})$. As this matrix is just a reduced version of $B(\mathcal{N})$, $B(\mathcal{U})\mathbf{v} = \mathbf{0}$ must hold.

We can see that $j \in \mathcal{P} \iff j \notin \bigcup_{k\in\mathcal{R}^*} \mathcal{O}(k)$ and $j \notin \bigcup_{k\in\mathcal{R}^*} \mathcal{I}(k)$. Therefore, $A(\mathcal{R}^*)\mathbf{v} = \mathbf{0}$. With these two results, $L(\mathcal{R}^*)\mathbf{v} = \mathbf{0}$. □

Lemma 6. *Let $k \in \mathcal{R}^*$ such that $e_0 \in \mathcal{I}(k)$ and $\mathcal{O}(k) = \{e_1, e_2, \ldots, e_q\}$. Let $\mathcal{P}_i = \{e_i, \ldots, e_0\}$ for $i = 1, \ldots, q$ be a feasible path, with associated path vector $\mathbf{v}_i$, such that it does not include any node from $\mathcal{R}^*$ other than k. Then $\mathbf{v} = \sum_i r_{e_0,e_i}\mathbf{v}_i$ belongs to the null space of $L(\mathcal{R}^*)$.*

Proof. Note that each of the paths $\mathcal{P}_i$ is a cycle: the path ends with e_0 which is connected via k to the starting edge e_i. We have $B(\mathcal{U})\mathbf{v}_i = \mathbf{0}$. Also, $A(\mathcal{R}^* \setminus \{k\})\mathbf{v}_i = \mathbf{0}$, as there are no shared edges between $\mathcal{P}_i$ and $\bigcup_{h\in\mathcal{R}^*, h\neq k} \mathcal{O}(h)$ or $\bigcup_{h\in\mathcal{R}^*, h\neq k} \mathcal{I}(h)$. It can be seen that

$$A(\{k\})\mathbf{v}_i = \mathbf{u}_i - \begin{bmatrix} r_{e_0,e_1} \\ r_{e_0,e_2} \\ \vdots \\ r_{e_0,e_q} \end{bmatrix}.$$

Thus,

$$\begin{aligned} A(\{k\})\mathbf{v} &= \sum_{i=1}^{q} r_{e_0,e_i} A(\{k\})\mathbf{v}_i \\ &= \sum_{i=1}^{q} r_{e_0,e_i}\mathbf{u}_i - \begin{bmatrix} r_{e_0,e_1} \\ r_{e_0,e_2} \\ \vdots \\ r_{e_0,e_q} \end{bmatrix} \sum_{i=1}^{q} r_{e_0,e_i} \\ &= \begin{bmatrix} r_{e_0,e_1} \\ r_{e_0,e_2} \\ \vdots \\ r_{e_0,e_q} \end{bmatrix} - \begin{bmatrix} r_{e_0,e_1} \\ r_{e_0,e_2} \\ \vdots \\ r_{e_0,e_q} \end{bmatrix} \\ A(\{k\})\mathbf{v} &= \mathbf{0} \end{aligned} .$$

Thus, $A(\mathcal{R}^*)\mathbf{v} = \mathbf{0}$, which implies that $L(\mathcal{R}^*)\mathbf{v} = \mathbf{0}$ finalizing the proof. □

In general, for each $j \in \mathcal{I}(k)$ for some $k \in \mathcal{R}^*$, the cycles that connect every outgoing edge of k to j must appear together in the elements of $\ker L(\mathcal{R}^*)$.

Theorem 3. *The rows of $C(\mathcal{S})$ calculated via the described algorithm and the rows of $L(\mathcal{R}^*)$ form a linearly independent set.*

Proof. The algorithm generates a set $\mathcal{S}$ of sensor locations such that this edges do not belong to a constructed spanning tree or a set of removed edges. It is well known that adding an edge $s \in \mathcal{S}$ to the tree generates a single cycle $\mathcal{P}_s$. Additionally, if the corresponding path vector to $\mathcal{P}_s$ is $\mathbf{v}_s$, the collection of $\mathbf{v}_s$ for $s \in \mathcal{S}$ is a linearly independent set. Consider a partition of $\mathcal{S}$ in $\mathcal{S}_\mathcal{U}$ and $\mathcal{S}_\mathcal{R}$, such that $\mathcal{S}_\mathcal{U}$ consists of the edges that when added to the tree will generate cycles that do not include any node from $\mathcal{R}^*$.

Consider one element $s_0 \in \mathcal{S}_\mathcal{U}$. From Lemma 5, $\mathbf{v}_{s_0} \in \ker L(\mathcal{R}^*)$. As $s_0 \in \mathcal{P}_{s_0}$ then $\mathbf{v}_{s_0}(s_0) \neq 0$, therefore $\mathbf{u}_{s_0}^T$ is linearly independent to the rows of $L(\mathcal{R}^*)$. Now define $L' = [L(\mathcal{R}^*)^T \quad \mathbf{u}_{s_0}]^T$. Consider another $s_1 \in \mathcal{S}_U$, $s_0 \neq s_1$. Because $s_1 \notin \mathcal{P}_{s_0}$, it is clear that $\mathbf{u}_{s_0}(s_1) = 0$, hence $\mathbf{v}_{s_1} \in \ker L'$. Following the previous reasoning we obtain that $\mathbf{u}_{s_1}^T$ is linearly independent to the rows of L'. Repeating this process iteratively, it can be seen that the collection of $\mathbf{u}_s^T$ for all $s \in \mathcal{S}_\mathcal{U}$ is linearly independent to the rows of $L(\mathcal{R}^*)$.

Now consider one element $s_0 \in \mathcal{S}_\mathcal{R}$ which forms a cycle $\mathcal{P}_{s_0}$ that includes some $k \in \mathcal{R}^*$ when added to the tree. From Lemma 6, there is at least one vector $\mathbf{v}$ in the null space of $L(\mathcal{R})$ such that $\mathbf{v}(s_0) \neq 0$, so $\mathbf{u}_{s_0}^T$ is linearly independent to $L(\mathcal{R}^*)$. Consider another $s_1 \in \mathcal{S}_\mathcal{R}$ associated with cycle $\mathcal{P}_{s_1}$ such that $\mathcal{P}_{s_1}$ also includes node k. If there exist $i \in O(k) \cap \mathcal{P}_{s_0}$ and $j \in O(k) \cap \mathcal{P}_{s_1}$ with $i \neq j$, then Lemma 6 implies that the path vectors $\mathbf{v}_{s_0}, \mathbf{v}_{s_1}$ form part of the same $\mathbf{v} \in \ker L(\mathcal{R}^*)$. Therefore, $\{\mathbf{u}_{s_0}^T, \mathbf{u}_{s_1}^T\}$ and the rows of $L(\mathcal{R}^*)$ do not form a linearly independent set.

Nevertheless, step 2 of the algorithm removes all but one of the outgoing edges for the intersections in $\mathcal{R}^*$, implying that it is not possible to form two cycles $\mathcal{P}_{s_0}, \mathcal{P}_{s_1}$ that include two different outgoing edges from the same $k \in \mathcal{R}^*$ by only adding elements from $\mathcal{S}$ to the tree. Thus, all the path vectors $\mathbf{v}_s$ for all $s \in \mathcal{S}_{\mathcal{R}}$ are related to linearly independent vectors in the null space of $L(\mathcal{R}^*)$. Hence, the collection of $\mathbf{u}_s^T$ for all $s \in \mathcal{S}_{\mathcal{R}}$ are linearly independent to the rows of $L(\mathcal{R}^*)$.

Furthermore, note that as the cycles generated by the elements of $\mathcal{S}_{\mathcal{U}}$ do not include any node from $\mathcal{R}^*$, Lemma 6 does not apply to them so they are independent to the cycles formed by the elements of $\mathcal{S}_{\mathcal{R}}$. Thus, the the rows of $C(\mathcal{S})$ and the rows of $L(\mathcal{R}^*)$ form a linearly independent set. □

References

Abbott-jard, M., Shah, H., Bhaskar, A., 2013. Empirical evaluation of Bluetooth and Wifi scanning for road transport. Proceedings of the Australasian Transport Research Forum (October), 1–14.

Bhaskar, A., Qu, M., Chung, E., 2015. Bluetooth vehicle trajectory by fusing bluetooth and loops: Motorway travel time statistics. IEEE Transactions on Intelligent Transportation Systems 16 (1), 113–122.

Bianco, L., Confessore, G., Gentili, M., 2006. Combinatorial aspects of the sensor location problem. Annals of Operations Research 144 (1), 201–234.

Bianco, L., Confessore, G., Reverberi, P., 2001. A Network Based Model for Traffic Sensor Location with Implications on O/D Matrix Estimates. Transportation Science 35 (1), 50–60.

Castillo, E., Calviño, A., Lo, H. K., Menéndez, J. M., Grande, Z., 2014. Nonplanar hole-generated networks and link flow observability based on link counters. Transportation Research Part B: Methodological 68, 239–261.

Castillo, E., Calvino, A., Menendez, J. M., Jimenez, P., Rivas, A., 2013. Deriving the upper bound of the number of sensors required to know all link flows in a traffic network. IEEE Transactions on Intelligent Transportation Systems 14 (2), 761–771.

Castillo, E., Grande, Z., Calviño, A., Szeto, W. Y., Lo, H. K., 2015. A State-of-The-Art Review of the Sensor Location, Flow Observability, Estimation, and Prediction Problems in Traffic Networks. Journal of Sensors 2015.

Castillo, E., Jiménez, P., Menéndez, J. M., Conejo, A. J., 2008. The observability problem in traffic models: Algebraic and topological methods. IEEE Transactions on Intelligent Transportation Systems 9 (2), 275–287.

Daganzo, C. F., 1995. The cell transmission model, part II: Network traffic. Transportation Research Part B 29 (2), 79–93.

Fu, C., Zhu, N., Ling, S., Ma, S., Huang, Y., 2016. Heterogeneous sensor location model for path reconstruction. Transportation Research Part B: Methodological 91, 77–97.

Gentili, M., Mirchandani, P. B., 2012. Locating sensors on traffic networks: Models, challenges and research opportunities. Transportation Research Part C: Emerging Technologies 24, 227–255.

He, S. X., 2013. A graphical approach to identify sensor locations for link flow inference. Transportation Research Part B: Methodological 51 (516), 65–76.

Hu, S. R., Liou, H. T., 2014. A generalized sensor location model for the estimation of network origin-destination matrices. Transportation Research Part C: Emerging Technologies 40, 93–110.

Hu, S. R., Peeta, S., Chu, C. H., 2009. Identification of vehicle sensor locations for link-based network traffic applications. Transportation Research Part B: Methodological 43 (8-9), 873–894.

Hu, S. R., Peeta, S., Liou, H. T., 2016. Integrated determination of network origin-destination trip matrix and heterogeneous sensor selection and location strategy. IEEE Transactions on Intelligent Transportation Systems 17 (1), 195–205.

Lovisari, E., Canudas de Wit, C., Kibangou, A. Y., 2016. Density/Flow reconstruction via heterogeneous sources and Optimal Sensor Placement in road networks. Transportation Research Part C: Emerging Technologies 69, 451–476.

Ng, M. W., 2012. Synergistic sensor location for link flow inference without path enumeration: A node-based approach. Transportation Research Part B: Methodological 46 (6), 781–788.

Rinaldi, M., Viti, F., 2017. Exact and approximate route set generation for resilient partial observability in sensor location problems. Transportation Research Part B: Methodological 105, 86–119.

Shivakumar, P. N., Chew, K. H., mar 1974. A Sufficient Condition for Nonvanishing of Determinants. Proceedings of the American Mathematical Society 43 (1), 63.

Viti, F., Rinaldi, M., Corman, F., Tampère, C. M., 2014. Assessing partial observability in network sensor location problems. Transportation Research Part B: Methodological 70, 65–89.

Xu, X., Lo, H. K., Chen, A., Castillo, E., 2016. Robust network sensor location for complete link flow observability under uncertainty. Transportation Research Part B: Methodological 88, 1–20.