https://doi.org/10.1016/j.ejor.2015.05.070

Vehicle-ID Sensor Location for Route Flow Recognition: Models and Algorithms

C. Cerrone*, R. Cerulli†, M. Gentili ‡

1 Introduction

The problem of locating sensors on the network to determine traffic flow volumes has been object in the past few years of growing interest. Problems in this class differentiate according to the type of sensors that need to be located (counting sensors, path-ID sensors, vehicle-ID sensors or a combination of them) and flows of interest (origin/destination flow volumes, arc flow volumes, route flow volumes, or a combination of them). Following the classification defined in the recent survey by Gentili and Mirchandani [15], two main classes of problems can be identified in this context: (i) locating sensors to fully observe flow volumes on the network (Sensor Location Flow-Observability Problems), and (ii) locating sensors to estimate flow volumes on the network (Sensor Location Flow-Estimation Problems). This division derives from the observation that the location of sensors on a network (either on the nodes or on the links) can be translated into a system of linear equations where the set of variables corresponds to the unknown flows and the set of equations comes from the deployed sensors. When the resulting system has a unique solution, we say the system is fully observable and therefore, under the assumption of error free data, all the flows involved in the system are known (that is they are *observable*). The resulting location problem consists in determining the optimum deployment of sensors on the network that results in an observable system (Sensor Location Flow-Observability Problems) and therefore it allows to determine all the flow volumes of interest. Observability problems arise, for example, when either path-ID sensors or vehicle-ID sensors need to be located on the links of the network to determine route flow volumes or link flow volumes (Gentili and Mirchandani [14],[15],[16], Hu et al. [17], He [25], Castillo et al. [2],[3],[4],[5],[6], [7], Minguez et al. [22], Ng [23]). On the other hand, when the system is not observable (that is, it is underspecified),

*University of Salerno, Department of Mathematics. Via Giovanni Paolo II, 132 - 84084 - Fisciano (SA), Italy. Email: cerronecarmine@gmail.com

†University of Salerno, Department of Mathematics. Via Giovanni Paolo II, 132 - 84084 - Fisciano (SA), Italy. Email: raffaele@unisa.it

‡University of Salerno, Department of Mathematics. Via Giovanni Paolo II, 132 - 84084 - Fisciano (SA), Italy. Email: mgentili@unisa.it

it admits an infinite number of solutions. The related location problem consists in determining how to optimally deploy sensors on the network so that the derived flow estimates are the best possible. Generally, underspecified systems arise when one is interested in determining origin-destination flow volumes by locating counting sensors on the links of the network or when there is a budget constraint that limits the number of sensors that can be located. This problem has been hugely studied in the literature (see for example, Chootinan et al. [8], Eisenman et al.[10], Elhert et al. [11], Fei and Mahmassani [12], Yang et al. [18] Kim et al. [19], Lam and Lo [20], Li and Ouyang [21], Yang and Zhou [26], Zhou and List [27], Fei et al. [13], Simonelli et al. [24]).

In this paper we focus on both the observability and the estimation problems arising when vehicle-ID sensors are to be located on the links of the network to determine and/or estimate route flow volumes. Both the problems were defined in Castillo et al. [9] and Mínguez et al. [22] respectively, where the corresponding mathematical formulations were presented and some computational experiments were carried out to evaluate the quality of the resulting observed/estimated route flows. The experiments were performed by solving the proposed mathematical formulations by means of a solver. Since the problems are NP-hard only small networks were tested.

We further study the two problems and provide contributions both from a theoretical and an algorithmic point of view. In particular, we improve the existing mathematical formulations by adding some new constraints that better define the feasible set of solutions, we also provide efficient solution approaches, still missing in the literature, namely two different greedy approaches and a Tabu Search algorithm. The efficiency of the proposed approaches is tested on a set of benchmark tests by comparing the provided solutions with the optimal ones, when available, returned by the solver CPLEX.

The sequel of the paper is organized as follows. Next section describes the addressed problems. Section 3 contains the mathematical formulations of the problems and the existing results in the literature. Section 4 describes our solution approaches. Computational results are discussed in Section 5. Conclusions and further research are object of Section 6.

2 The Vehicle-ID Location Problem: Observability and Estimation

In this section we formally describe the addressed problems. We follow the general description given in [15] to introduce the Sensor Location Flow-Observability Problem and the Sensor Location Flow-Estimation

OD pair	***OD trips***	***Routes***	***Route Flow***
$w_1 = (1,5)$	27	R_1 : a_1 a_2 a_3 a_4	15
		R_2 : a_1 a_7 a_4	12
$w_2 = (1,4)$	10	R_3 : a_1 a_6 a_8 a_3	10
$w_3 = (3,2)$	7	R_4 : a_3 a_4 a_5 a_1	7
$w_4 = (4,3)$	22	R_5 : a_4 a_5 a_1 a_2	22

Links	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
Link Flow	66	37	32	56	29	10	12	10

Table 1: *OD Trips, Route Flows, Link Flows related to the network in Figure 1*

Problem when vehicle-ID sensors are to be located on the links of a network.

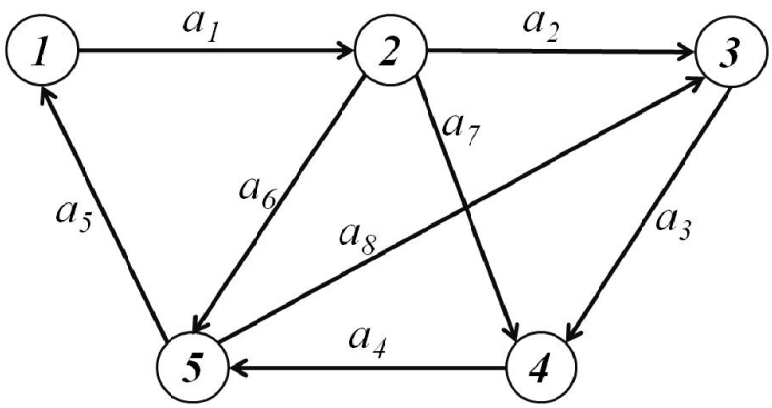

Figure 1: *An example network with 5 nodes and 8 links.*

Consider the network in Figure 1 with 5 nodes and 8 links, and assume there are 4 different OD pairs on the network each connected by one or more routes, namely: the OD pair $w_1 = (1,5)$ is connected by route $R_1 = \{a_1, a_2, a_3, a_4\}$ and by route $R_2 = \{a_1, a_7, a_4\}$, the OD pair $w_2 = (1,4)$ is connected by route $R_3 = \{a_1, a_6, a_8, a_3\}$, OD pair $w_3 = (3,2)$ is connected by route $R_4 = \{a_3, a_4, a_5, a_1\}$ and the OD pair $w_4 = (4,3)$ is connected by route $R_5 = \{a_4, a_5, a_1, a_2\}$. Table 1 lists all the OD pairs, the corresponding OD trips, all the 5 routes, the corresponding route flow volumes, and all the link flow volumes. For example, the OD trips corresponding to OD pair w_1 are equal to 27 units, 15 of them use route R_1 and the remaining vehicles use route R_2; flow on link a_2 is equal to 37 units and it is equal to the sum of

the flow of the routes that use that link, that is routes R_1 with flow 15 and route R_5 with flow equal to 22. The information shown in the table are given here for clarity of the exposition; we assume to know only the set of routes, while all the other information, namely, OD trips, link flows, and route flows are not known. Our aim is to locate Vehicle-ID sensors on the links of the network to derive the route flow volumes (listed in Table 1) that are not known. The location of a vehicle-ID sensor on a link of a network allows to univocally recognize a vehicle traveling on the network. When vehicle-ID sensors are located on the links we obtain flow volumes of the corresponding links. However, not only flows of the links can be obtained, but also, more detailed information can be derived when the same vehicle is detected on different links at different times. For example, let us suppose to locate three vehicle-ID sensors on our network example, say on links a_1, a_3 and a_4 (we say that links a_1, a_3 and a_4 are *scanned*). Since link a_1 is used by all the routes in the network, then the following equation can be derived:

$$(link\ \ a_1)\ \ f_{R_1} + f_{R_2} + f_{R_3} + f_{R_4} + f_{R_5} = 66 \tag{1}$$

where 66 is the intercepted flow on link a_1, and f_{R_i} denotes the unknown route flow volume of route R_i. Similarly, for links a_3 and a_4, we can derive the two following equations:

$$\begin{array}{lrcl} (link\ \ a_3) & f_{R_1} + f_{R_3} + f_{R_4} & = & 32 \\ (link\ \ a_4) & f_{R_1} + f_{R_2} + f_{R_4} + f_{R_5} & = & 56 \end{array} \tag{2}$$

Moreover, if we consider jointly links a_1 and a_4, the following two additional equations related to the number of vehicles that are recognized on *both* links a_1 and a_4, can also be obtained:

$$\begin{array}{lrcl} (links\ \ (a_1, a_4)) & f_{R_1} + f_{R_2} & = & 27 \\ (links\ \ (a_4, a_1)) & f_{R_4} + f_{R_5} & = & 29 \end{array} \tag{3}$$

Indeed, when a vehicle is detected, the detection time is also known. Hence, if the same vehicle is detected first on link a_1 and subsequently on link a_4, then we can state that such a vehicle is using any route that contains both the links in such an order, in this case the two routes R_1 and R_2. While if a vehicle is first intercepted on link a_4 and subsequently on link a_1 then we can say it is using either route R_4 or route R_5. We refer to this type of equation as the *joint recognition* equation associated with a subset of vehicle-ID sensors. Considering all the possible equations associated with the location set $U = \{a_1, a_3, a_4\}$, the

following system of linear equations will result:

$$\begin{array}{rcl} (link\ \ a_1) & f_{R_1}+f_{R_2}+f_{R_3}+f_{R_4}+f_{R_5} & = \ 66 \\ (link\ \ a_3) & f_{R_1}+f_{R_3}+f_{R_4} & = \ 32 \\ (link\ \ a_4) & f_{R_1}+f_{R_2}+f_{R_4}+f_{R_5} & = \ 56 \\ (links\ \ (a_1,a_4) & f_{R_1}+f_{R_2} & = \ 27 \\ (links\ \ (a_4,a_1)) & f_{R_4}+f_{R_5} & = \ 29 \\ (links\ \ (a_1,a_3)) & f_{R_1}+f_{R_3} & = \ 25 \\ (links\ \ (a_3,a_1)) & f_{R_4} & = \ 7 \\ (links\ \ (a_3,a_4)) & f_{R_1}+f_{R_4} & = \ 22 \\ (links\ \ (a_1,a_3,a_4)) & f_{R_1} & = \ 15 \\ (links\ \ (a_3,a_4,a_1)) & f_{R_4} & = \ 7. \end{array} \tag{4}$$

These equations are those associated with some of the possible ordered subsets of these three sensors. In particular, the first three equations are associated each with a single sensor, the subsequent five equations are the joint recognition equations associated with the possible subsets of two sensors out of the three that are located, and, the remaining two equations are the joint recognition equations associated with the three sensors together. Note that, not all the possible ordered subsets are considered, indeed there does not exist any route in the network that contains the ordered pair (a_4, a_3). The same observation can be made for the ordered sets containing all the three sensors: (a_1, a_4, a_3), (a_3, a_1, a_4), (a_4, a_1, a_3) and (a_4, a_3, a_1).

System (4) has 5 unknown variables and 5 linear independent equations, hence the system admits a unique solution and all route flow volumes f_{R_i}, $i = 1, \ldots, 5$ can be univocally determined. We say, in this case, that the location set $U = \{a_1, a_3, a_4\}$ has associated a fully observable system of linear equations. In this context, one could wonder whether there exists another subset of three or less sensors such that the associated system of equations is fully observable and hence allows to determine all route flow volumes. The following location problem arises:

Vehicle-ID Sensor Location Flow-Observability Problem:

What is the minimum number of vehicle-ID sensors to be located on the links of the network and where to locate them, such that the associated system of linear equations is fully observable, that is it allows to uniquely determine all route flow volumes?

A different question arises when a limited number of sensors, say k, can be located on the network and all the systems associated with all the possible subsets of k sensors do not admit a unique solution.

Indeed, in this case, how can one decide when a location set is better than another? A possible way is (see [15]) to define a location-evaluation criterion g that associates with each location set U a value $g(U)$ representing a measure of the quality of the flow estimates obtained by the system of equations associated with U. Given such a criterion, we can conclude that a location set U is better than another location set U' if the value $g(U)$ associated with U is better than the value $g(U')$ associated with U'. Following the approach suggested in [22], for the case where vehicle-ID sensors need to be located, we could associate with the location set U an evaluation criterion $g(U)$ that equals the total number of routes that can be univocally observed through the system associated with U. In this way, we could conclude that between two location sets U and U', U is better than U' if $g(U) > g(U')$. In our network example, let us consider the two locations sets $U_1 = \{a_1, a_4\}$ and $U_2 = \{a_3, a_5\}$. The following system is associated with the location set U_1:

$$\begin{array}{rcl} (link\ \ a_1) & f_{R_1} + f_{R_2} + f_{R_3} + f_{R_4} + f_{R_5} = 66 \\ (link\ \ a_4) & f_{R_1} + f_{R_2} + f_{R_4} + f_{R_5} = 56 \\ (links\ \ (a_1, a_4)) & f_{R_1} + f_{R_2} = 27 \\ (links\ \ (a_4, a_1)) & f_{R_4} + f_{R_5} = 29 \end{array} \tag{5}$$

and the following system is associated with the location set U_2:

$$\begin{array}{rcl} (link\ \ a_3) & f_{R_1} + f_{R_3} + f_{R_4} = 32 \\ (link\ \ a_5) & f_{R_4} + f_{R_5} = 29 \\ (links\ \ (a_3, a_5)) & f_{R_4} = 7 \end{array} \tag{6}$$

By solving system (5) route R_3 can be univocally determined and $g(U_1) = 1$, while by solving system (6) we have $g(U_2) = 2$ since routes R_4 and R_5 can be univocally determined. Hence, we could conclude that, according to the defined evaluation criterion g, U_2 is better than U_1. Hence, given an evaluation function g, the following location problem can be defined:

Vehicle-ID Sensor Location Flow-Estimation Problem:
If a given number k of vehicle-ID sensors are to be located on the network, where to locate them so that the associated evaluation criterion g is optimized?

Both the described problems are NP-hard as it will be evident in the next section where the mathematical formulation is presented and the relationship with the set-covering problem is outlined.

3 Mathematical formulations

In this section we provide the mathematical formulation of the two problems described in the previous section. To this aim we introduce the concept of *scanning map* (firstly defined in [9]) associated with a location set U. Let us return to our example network. Consider again the location set $U_1 = \{a_1, a_4\}$ and let us associate with each route R_i the set of links C_{R_i} of scanned links of the route (referred to in the sequel as *scanning set* associated with the route). That is: $C_{R_1} = (a_1, a_4)$, $C_{R_2} = (a_1, a_4)$, $C_{R_3} = (a_1)$, $C_{R_4} = (a_4, a_1)$ and $C_{R_5} = (a_4, a_1)$. Let us refer to this collection of sets as the *scanning map* associated with the location $\{a_1, a_4\}$. Analyzing this scanning map we can conclude: (i) if a vehicle is intercepted only on link a_1, then it is using route R_3; (ii) if a vehicle is intercepted first on link a_1 and later on link a_4 then it is using either route R_1 or R_2; finally, (iii) if a vehicle is intercepted first on link a_4 and later on link a_1 then it is using either route R_4 or R_5. That is, this scanning map allows to recognize univocally only vehicles of routes R_3, while for the remaining routes the same conclusion cannot be derived. Let us consider to locate one additional vehicle-ID sensor on link a_3, the new scanning map is: $C_{R_1} = (a_1, a_3, a_4)$, $C_{R_2} = (a_1, a_4)$, $C_{R_3} = (a_1, a_3)$, $C_{R_4} = (a_3, a_4, a_1)$ and $C_{R_5} = (a_4, a_1)$. The location of a vehicle-ID sensor on link a_3 permits to distinguish among those vehicles that use route R_1 and those vehicles using route R_2, and, similarly, we can distinguish those vehicles that use route R_4 and those vehicles that use route R_3. From this example, it is clear that, by using the scanning map associated with a location set, it is possible to understand which route, among all, is uniquely identified. Based on the scanning map definition we can give an alternative definition of the two problems described in the previous section, the mathematical formulation presented in [9] and [22] is based on these definitions.

Formally, let $G = (V, A)$ be a network and R be a set of routes defined on the network. Given a location of vehicle-ID sensors on a subset $U \subseteq A$ of links, we can define a scanning map $SM(U) = \{C_{R_i} : C_{R_i} = U \cap R_i, i = 1, 2, \ldots, |R|\}$, where each scanning set C_{R_i} is associated with route R_i and contains the scanned ordered list of links of the route. If all the scanning sets in the scanning map are not empty and are different, i.e. $C_{R_i} \neq C_{R_j}$, for each $i \neq j = 1, 2, \ldots, |R|$, then the scanning map allows to determine all the route flow volumes and we can define the alternative version of the observability problem given in the previous section:

Vehicle-ID Sensor Location Flow-Observability Problem:

What is the minimum number of vehicle-ID sensors to be located on the links of a network and where to locate them such that the associated scanning map allows observability of all route flow volumes?

while, if we define the evaluation criterion $g(U)$ as the number of routes that can be univocally determined through the scanning map associated with U, we can state the following alternative version of

the estimation problem:

Vehicle-ID Sensor Location Flow-Estimation Problem:

If a given number k of vehicle-ID sensors are to be located on the network, where to locate them so that the associated scanning map is such that the total number of univocally determined routes is maximized?

For a matter of clarity of exposition, in the sequel we will refer to the Vehicle-ID Sensor Location Flow-Observability Problem as problem P1, and, to the Vehicle-ID Sensor Location Flow-Estimation Problem a problem P2.

Let us introduce two sets of binary variables: variable x_a associated with each link $a \in A$ that it is equal to 1 if a vehicle-ID sensor is located on link a and it is equal to 0 otherwise; variable y_{ab} associated with each pair of links $a \neq b \in A$ that is equal to 1 if a vehicle-ID sensor is located both on link a and link b and it is equal to 0 otherwise. The following set of parameters are also defined:

- $\rho^a_{R_i}$: is equal to 1 if route R_i contains link a and is equal to 0 otherwise;
- $\rho^a_{R_iR_j}$: is equal to 1 if link a is contained either in R_i or in R_j (not in both) and is equal to 0 otherwise;
- $\sigma^{ab}_{R_iR_j}$: is equal to 1 if links a and b are both in route R_i and R_j but they appear in a different order.

The mathematical formulation of problem P1, is the following [**P1**]:

$$\min \sum_{a \in A} x_a \tag{7}$$

$$s.t. \sum_{a \in A} \rho^a_{R_i} x_a \geq 1 \qquad \forall\, R_i \in R \tag{8}$$

$$\sum_{a \in A} \rho^a_{R_iR_j} x_a + \sum_{a \neq b \in A} \sigma^{ab}_{R_iR_j} y_{ab} \geq 1 \qquad \forall\, (R_i, R_j) | R_i < R_j \tag{9}$$

$$2y_{ab} - x_a - x_b \leq 0 \qquad \forall\, a \neq b \in A \tag{10}$$

$$x_a \in \{0,1\} \qquad \forall\, a \in A \tag{11}$$

$$y_{ab} \in \{0,1\} \qquad \forall\, a \neq b \in A \tag{12}$$

The objective function (7) is the sum of the total number of the vehicle-ID sensors located on the network and is minimized. We refer to constraints (8) as *covering constraints* as they ensure there is at least one vehicle-ID sensor located on each route of the network. Constraints (9) are *diversification constraints*. These constraints ensure the scanning map associated with the location of vehicle-ID sensors allows full observability of the route flow volumes. Indeed, the sum on the left of the constraint ensures that for each couple of routes R_i and R_j the corresponding associated subsets C_{R_i} and C_{R_j} are different either

because they contain at least one different link or, in case they contain the same set of links, they appear in different order. Constraints (10) are logical constraints linking the binary variables x_a and y_{ab}. If y_{ab} is equal to 1 then the constraint forces both x_a and x_b to be equal to 1, otherwise, if $y_{ab} = 0$ the minimization of the objective function drives the value of x_a and x_b to be equal to 0. The remaining constraints (11) and (12) require the variables to be binary.

The above mathematical formulation is different from the one provided in [9] where the set of diversification constraints did not take into account the order the scanned links appear in each set C_{R_i}. Our diversification constraints (9)-(10) consider this order and hence the feasible set of solutions of the problem is now correctly described. In Section 5 the results of the two mathematical formulations are compared on different sets of instances. The results show that our modified version of the diversification constraints improves the final solution on those instances where the routes in the network are defined as subsets of circular paths like, for example, the two routes R_1 and R_4 in the network of Figure 1 that are subsets of the circular path $P = \{a_1, a_2, a_3, a_4, a_5\}$.

Let us consider the additional set of binary variables z_{R_i} associated with each route R_i, such that z_{R_i} is equal to 1 if route R_i is uniquely identified (that is observed) and it is equal to 0 otherwise. Let us also define a weight w_{R_i} associated with route R_i to define its relative importance (see for example [22] for possible definitions of these weights). The mathematical formulation of problem P2 is then the following **[P2]**:

$$\max \sum_{R_i \in R} w_{R_i} z_{R_i} \tag{13}$$

$$s.t. \sum_{a \in A} \rho^a_{R_i} x_a \geq z_{R_i} \qquad \forall\, R_i \in R \tag{14}$$

$$\sum_{a \in A} \rho^a_{R_i R_j} x_a + \sum_{a \neq b \in A} \sigma^{ab}_{R_i R_j} y_{ab} \geq z_{R_i} \qquad \forall\, (R_i, R_j) | R_i < R_j \tag{15}$$

$$2y_{ab} - x_a - x_b \leq 0 \qquad \forall\, a \neq b \in A \tag{16}$$

$$\sum_{a \in A} x_a \leq k \tag{17}$$

$$x_a \in \{0,1\} \qquad \forall\, a \in A \tag{18}$$

$$y_{ab} \in \{0,1\} \qquad \forall\, a \neq b \in A \tag{19}$$

$$z_{R_i} \in \{0,1\} \qquad \forall\, R_i \in R \tag{20}$$

The objective function (22) maximizes the total weight of the routes that can be univocally determined. Constraints (14) and (15) differ from those in the previous model [P1] since they ensure the coverage and the diversification of route R_i only if the corresponding variable z_{R_i} is equal to 1. Constraints (16) are

the same as in the previous model. Constraint (17) ensures at most k vehicle-ID sensors are located on the links of the network. Variables are ensured to be binary by constraints (18) - (20). Note that, by fixing $w_{R_i} = 1$, $\forall R_i \in R$, the problem looks for the maximization of the total number of routes that can be univocally determined.

As for the previous formulation [P1], the above formulation of the problem differs from the one provided in [22] basically for the characterization of the diversification constraints (15)-(16) that, unlike the constraints defined in [22], take indirectly into account the order the scanned links appear in the scanning sets.

In the sequel of the paper, we consider also two weighted variants of the above described problems that arise when an installation cost c_a is associated with each link. In particular, a variant of problem [P1] requires to determine the minimum cost vehicle-ID sensor location that ensures the observability of all the routes in the network. The corresponding model formulation is the following [**P1b**]

$$\min \sum_{a \in A} c_a x_a \tag{21}$$

$$\text{s.t. } (8)\text{-}(12)$$

If there is an available maximum budget C_{max} for the location of the sensors, then one could be interested in locating vehicle-ID sensors on the links of the network to maximize the total weight of the routes that can be univocally determined without exceeding the budget limit. In this case, the following model describes this variant [**P2b**]:

$$\max \sum_{R_i \in R} w_{R_i} z_{R_i} \tag{22}$$

$$\text{s.t.} \quad \sum_{a \in A} c_a x_a \leq C_{max} \tag{23}$$

$$(14) - (16), (18) - (20) \tag{24}$$

4 Solution approaches

In this section we present our solution approaches, namely two greedy approaches (Greedy1 and Greedy2) and a Tabu Search algorithm, to solve problems P1, P1b, P2 and P2b. We describe the approaches for problem P1 and subsequently we will explain how the approaches have been modified to solve problems P1b, P2 and P2b.

(*covering* R_1)	x_{a_1}	$+x_{a_2}$	$+x_{a_3}$	$+x_{a_4}$							$\geqslant$	1
(*covering* R_2)	x_{a_1}			$+x_{a_4}$			$+x_{a_7}$				$\geqslant$	1
(*covering* R_3)	x_{a_1}		$+x_{a_3}$			$+x_{a_6}$		$+x_{a_8}$			$\geqslant$	1
(*covering* R_4)	x_{a_1}		$+x_{a_3}$	$+x_{a_4}$	$+x_{a_5}$						$\geqslant$	1
(*covering* R_5)	x_{a_1}	$+x_{a_2}$		$+x_{a_4}$	$+x_{a_5}$						$\geqslant$	1
(*diversification* R_1R_2)		x_{a_2}	$+x_{a_3}$				$+x_{a_7}$				$\geqslant$	1
(*diversification* R_1R_3)		x_{a_2}		$+x_{a_4}$		$+x_{a_6}$		$+x_{a_8}$			$\geqslant$	1
(*diversification* R_1R_4)		x_{a_2}			$+x_{a_5}$				$+y_{a_1a_3}$	$+y_{a_1a_4}$	$\geqslant$	1
(*diversification* R_1R_5)			x_{a_3}		$+x_{a_5}$					$+y_{a_1a_4}$ $+y_{a_2a_4}$	$\geqslant$	1
(*diversification* R_2R_3)			x_{a_3}	$+x_{a_4}$		$+x_{a_6}$	$+x_{a_7}$	$+x_{a_8}$			$\geqslant$	1
(*diversification* R_2R_4)			x_{a_3}		$+x_{a_5}$		$+x_{a_7}$			$+y_{a_1a_4}$	$\geqslant$	1
(*diversification* R_2R_5)		x_{a_2}			$+x_{a_5}$		$+x_{a_7}$			$+y_{a_1a_4}$	$\geqslant$	1
(*diversification* R_3R_4)				x_{a_4}	$+x_{a_5}$	$+x_{a_6}$		$+x_{a_8}$	$+y_{a_1a_3}$		$\geqslant$	1
(*diversification* R_3R_5)		x_{a_2}	$+x_{a_3}$	$+x_{a_4}$	$+x_{a_5}$	$+x_{a_6}$		$+x_{a_8}$			$\geqslant$	1
(*diversification* R_4R_5)		x_{a_2}	$+x_{a_3}$								$\geqslant$	1

Table 2: Set of Covering constraints and Diversification constraints associated with the example in Figure 1.

4.1 Greedy1 Approach

Before describing the greedy criterion used by our Greedy1 algorithm, let us introduce first some definitions and notations. Given a subset of links $U \subseteq A$, we associate with each link $a \in A$ two different scores according to the number of additional constraints that can be satisfied if a is selected. In particular we define the two following criteria:

- the **Covering Criterion**: $Cov(U, a)$ is the additional set of routes that can be covered if link a is selected (that is, the set of additional covering constraints that can be satisfied if link a is selected). The resulting score associated with link a is the size of this set, that is $|Cov(U, a)|$.
- the **Diversification Criterion**: $Div(U, a)$ is the set of additional diversification constraints that can be satisfied since either the term $\rho^a_{R_iR_j} x_a$, or the term $\sigma^{ab}_{R_iR_j} y_{ab}$, becomes equal to 1 in the summation of the constraints where x_a or y_{ab} are involved, if a is selected. The resulting score associated with link a is the size of this set, that is $|Div(U, a)|$.

Let us return to our example in Figure 1 for which the entire set of covering and diversification constraints of the mathematical formulation [$P1$] is given in Table 2. Let us assume link a_4 is selected, that is $U = \{a_4\}$, the corresponding value of the above criteria for each of the remaining links are listed in Table 3. The covering constraints that are already satisfied by the selection of link a_4 are those related to routes: R_1,R_2,R_4 and R_5, while the satisfied diversification constraints are those corresponding to the pairs: $\{R_1, R_3\}, \{R_2, R_3\}, \{R_3, R_4\}$ and $\{R_3, R_5\}$. Consider for example link a_2. Link a_2 has a covering score equal to $|Cov(\{a_4\}, a_2)| = 0$ since there is no additional covering constraint that is satisfied if a_2 is selected, while the diversification score is equal to 5; indeed, the constraints related the pairs $\{R_1, R_2\}, \{R_1, R_4\}, \{R_2, R_5\}$ and $\{R_4, R_5\}$ are satisfied since variable $x_{a_2} = 1$ and the constraint related to the pair $\{R_1, R_5\}$ is satisfied since variable $y_{a_2a_4} = 1$.

	$\lvert Cov(\{a_4\}, a_i)\rvert$	$\lvert Div(\{a_4\}, a_i)\rvert$
a_1	1	4
a_2	0	5
a_3	1	4
a_5	0	4
a_6	1	0
a_7	0	3
a_8	1	0

Table 3: Scores of each link of the network of Figure 1 for $U = \{a_4\}$.

According to the two criteria we define an initial ordering of the links by ordering them in decreasing order with respect to the Covering criterion for $U = \emptyset$; ties are broken according to a decreasing order of the Diversification criterion for $U = \emptyset$. We refer to this ranking as **Initial Ranking**. The *Initial Ranking* for our network example is: $\{a_1, a_4, a_3, a_5, a_2, a_6, a_7, a_8\}$ (see the upper most Table in Figure 2 where the values of the Covering criterion and of the Diversification criterion are showed for our example when $U = \emptyset$).

Our first greedy approach to solve problem P1, iteratively selects a link that maximizes the covering criterion; in case of ties the link that maximizes the diversification criterion is chosen; in case of ties the link having the best position in the *Initial Ranking* is selected. The algorithm stops when a feasible solution is obtained. A re-optimization step is also applied to the returned solution U. This step tries to eliminate redundant links. The details of this algorithm are given in Table 4. In the worst case, the running time required by this algorithm is $O(|A||R|^2)$; indeed in the worst case, the algorithm could, for each link, check the entire set of constraints.
Figure 2 shows the steps of Greedy1 when it is applied to solve problem P1 on our example network of Figure 1.

Greedy1 - P1

1. `Initialization`
   Let $U \leftarrow \emptyset$ be the set of selected links.
   Let A be the set of links in the network.
2. `Greedy step`
   2.1 While U is not a feasible solution
      find the link $a \in A \backslash U$ with $|Cov(U, a)|$ maximum;
      in case of ties select the one with $|Div(U, a)|$ maximum;
      in case of ties select the one with the best position in the *Initial Ranking*
3. Return U

4. `Re-Optimization Step`
   $U = \{a_{i_1}, a_{i_2}, \ldots, a_{i_h}\}$;
   $j \leftarrow h$;
   While $j > 0$
      $U \leftarrow U \backslash \{a_{i_j}\}$;
      if U is not feasible then $U \leftarrow U \cup \{a_{i_j}\}$
      $j \leftarrow j - 1$
5. Return U

Table 4: *Greedy1 Algorithm to solve Problem P1.*

	$\lvert Cov(\emptyset, a_i)\rvert$	$\lvert Div(\emptyset, a_i)\rvert$
a_1	5	0
a_2	2	6
a_3	3	6
a_4	4	4
a_5	2	6
a_6	1	4
a_7	1	4
a_8	1	4

- Scores of each link of the network of Figure 1 for $U = \emptyset$.
- The resulting initial ranking is: $\{a_1, a_4, a_3, a_5, a_2, a_6, a_7, a_8\}$.
- The first selected link is a_1: $U = \{a_1\}$.

	$\lvert Cov(\{a_1\}, a_i)\rvert$	$\lvert Div(\{a_1\}, a_i)\rvert$
a_2	0	6
a_3	0	8
a_4	0	8
a_5	0	6
a_6	0	4
a_7	0	4
a_8	0	4

- The second selected link is a_4: $U = \{a_1, a_4\}$.

	$\lvert Cov(\{a_1, a_4\}, a_i)\rvert$	$\lvert Div(\{a_1, a_4\}, a_i)\rvert$
a_2	0	2
a_3	0	2
a_5	0	0
a_6	0	0
a_7	0	1
a_8	0	0

- The third selected link is a_3: $U = \{a_1, a_4, a_3\}$.
- The current set U is feasible.
- The re-optimization step does not eliminate any element from U.
- The final returned solution is optimal.

Figure 2: Steps of Greedy1 algorithm when it is applied to solve problem P1 on the network of Figure 1.

Greedy2 - P1

1. `Initialization`
   Let $U \leftarrow \emptyset$ be the set of selected links.
   Let A be the set of links in the network.
2. `Greedy step`
   2.1 While U is not a feasible solution
      Find the link $a \in A \backslash U$ such that $W(U, a)$ is maximum;
      Set $U \leftarrow \{a\}$;
3. Return U

Table 5: *Greedy2 Algorithm to solve Problem P1.*

4.2 Greedy2 Approach

Given a subset of links $U \subseteq A$, our second greedy approach to solve problem P1, iteratively selects a link a that maximizes a greedy criterion $W(U, a)$ that takes into account three main components: (i) length of the routes (in terms of number of links) that are covered if link a is selected, (ii) number of routes that can be univocally identified if link a is selected, and (iii) a measure of the total number of routes that still need to be differentiated if link a is selected. The algorithm stops when a feasible solution is obtained. The re-optimization step is applied to the returned solution U. The description of the greedy function is given next, while the pseudocode of the algorithm is given in Table 5. Let us better describe the function $W(U, a)$ that is the core of the greedy criterion at Step 2.1. Let R be the set of the routes on the given network and let $U \subseteq A$ be a subset of links. Function $W(U, a)$ favors the choice of those links that, on the one hand, improve most the feasibility of the solution, and, on the other hand, leave a set of routes that are easier to be differentiated. These aspects are taken into account by considering the sum of three weighted components:

$$W(U, a) = \rho_1 f_1(U, a) + \rho_2 f_2(U, a) + \rho_3 f_3(U, a)$$

The first $f_1(U, a)$ component is computed as:

$$f_1(U, a) = \sum_{R_i \in Cov(U,a)} \frac{Avg}{|R_i|}$$

where $|R_i|$ is the total number of links contained in route R_i, and Avg is the average number of links of each route $R_j \in R$ (for the example in Figure 1 we have $Avg = \frac{19}{5} = 3.8$). This criterion $f_1(U, a)$ takes into account the total number of routes that can be additionally covered if link a is selected. However, unlike the covering criterion, it also takes into account the length (in terms of number of links) of the

additional routes that can be covered. Indeed, routes with a fewer number of links are less likely to be covered, hence, function f_1 favors the selection of those links that cover shortest routes.
The second component $f_2(U, a)$ is the following:

$$f_2(U, a) = |D(U, a)|.$$

This component equals the total number of routes that can be univocally identified by the selection of link a. This criterion is slightly different from the diversification criterion introduced previously. Indeed, $Div(U, a)$ is the set of diversification constraints that can be satisfied when link a is selected, while $D(U, a)$ is the set of routes that are associated with a unique nonempty scanning set in the scanning map resulting after the selection of link a.
Finally, the last component is the following:

$$f_3(U, a) = \sum_{\substack{C_{R_i} \in SM(U \cup \{a\}): \\ R_i \notin D(U,a) \\ R_i \in Cov(\emptyset,a)}} \sqrt{|M(C_{R_i})|}.$$

This component considers the scanning map that results after the selection of link a. In particular, for each route that is covered by link a, i.e. $R_i \in Cov(\emptyset, a)$ and that is still not univocally determined, that is $R_i \notin D(U, a)$, the associated scanning set C_{R_i} is considered. The total number of routes whose scanning set is equal to C_{R_i}, that is $|M(C_{R_i})| = |\{R_j \in R : R_j \neq R_i \ \ and \ \ C_{R_j} \equiv C_{R_i}\}|$ is computed. Function f_3 computes the sum of the square root of the size of these sets. This criterion privileges the selection of those links that, if selected, split the resulting scanning map as much as possible.
These three functions are properly weighted with weights ρ_i, $i = 1, 2, 3$ to define the relative importance of the corresponding criterion. In our experiments we set $\rho_1 = 2\rho_2$, $\rho_2 = 100$ and $\rho_3 = 1$. We chose these values to favor criterion f_1 first, then criterion f_2, and finally to take into account criterion f_3 in case of ties.

Figure 3 shows the steps of Greedy2 when it is applied to solve problem P1 on the example of Figure 1. In the example we set $\rho_1 = \rho_2 = \rho_3 = 1$. Tables 6-8 show the scanning maps associated with each link and with sets $U = \emptyset$, $U = \{a_1\}$, and $U = \{a_1, a_3\}$ respectively.
In the worst case, the running time required by this algorithm is $O(|A||R|^2)$; indeed in the worst case, the algorithm could, for each link, check the entire set of constraints.

	$SM(\{a_1\})$	$SM(\{a_2\})$	$SM(\{a_3\})$	$SM(\{a_4\})$	$SM(\{a_5\})$	$SM(\{a_6\})$	$SM(\{a_7\})$	$SM(\{a_8\})$
R_1	(a_1)	(a_2)	(a_3)	(a_4)				
R_2	(a_1)			(a_4)			(a_7)	
R_3	(a_1)		(a_3)			(a_6)		(a_8)
R_4	(a_1)		(a_3)	(a_4)	(a_5)			
R_5	(a_1)	(a_2)		(a_4)	(a_5)			

Table 6: Scanning maps associated with the singleton sets $\{a_i\}$, $\forall a_i \in A$. These scanning maps define the scores of each link as shown in the upper table of Figure 3.

	$SM(\{a_1\}, a_2)$	$SM(\{a_1\}, a_3)$	$SM(\{a_1\}, a_4)$	$SM(\{a_1\}, a_5)$	$SM(\{a_1\}, a_6)$	$SM(\{a_1\}, a_7)$	$SM(\{a_1\}, a_8)$
R_1	(a_1, a_2)	(a_1, a_3)	(a_1, a_4)	(a_1)	(a_1)	(a_1)	(a_1)
R_2	(a_1)	(a_1)	(a_1, a_4)	(a_1)	(a_1)	(a_1, a_7)	(a_1)
R_3	(a_1)	(a_1, a_3)	(a_1)	(a_1)	(a_1, a_6)	(a_1)	(a_1, a_8)
R_4	(a_1)	(a_3, a_1)	(a_4, a_1)	(a_5, a_1)	(a_1)	(a_1)	(a_1)
R_5	(a_1, a_2)	(a_1, a_3)	(a_4, a_1)	(a_5, a_1)	(a_1)	(a_1)	(a_1)

Table 7: Scanning maps associated with the sets $\{\{a_1\}, a_i\}$, $\forall a_i \in A \backslash \{a_1\}$. These scanning maps define the scores of each link as shown in the middle table of Figure 3.

	$SM(\{a_1, a_4\}, a_2)$	$SM(\{a_1, a_4\}, a_3)$	$SM(\{a_1, a_4\}, a_5)$	$SM(\{a_1, a_4\}, a_6)$	$SM(\{a_1, a_4\}, a_7)$	$SM(\{a_1, a_4\}, a_8)$
R_1	(a_1, a_2, a_4)	(a_1, a_3, a_4)	(a_1, a_4)	(a_1, a_4)	(a_1, a_4)	(a_1, a_4)
R_2	(a_1, a_4)	(a_1, a_4)	(a_1, a_4)	(a_1, a_4)	(a_1, a_7, a_4)	(a_1, a_4)
R_3	(a_1)	(a_1, a_3)	(a_1)	(a_1, a_6)	(a_1)	(a_1, a_8)
R_4	(a_4, a_1)	(a_3, a_4, a_1)	(a_4, a_5, a_1)	(a_4, a_1)	(a_4, a_1)	(a_4, a_1)
R_5	(a_4, a_1, a_2)	(a_4, a_1)	(a_4, a_5, a_1)	(a_4, a_1)	(a_4, a_1)	(a_4, a_1)

Table 8: Scanning maps associated with the sets $\{\{a_1, a_3\}, a_i\}$, $\forall a_i \in A \backslash \{a_1, a_3\}$. These scanning maps define the scores of each link as shown in the last table of Figure 3.

	$f_1(\emptyset, a_i)$	$f_2(\emptyset, a_i)$	$f_3(\emptyset, a_i)$	$W(\emptyset, a_i)$
a_1	5.07	0	$\sqrt{5}$	$5.07 + \sqrt{5}$
a_2	1.90	0	$\sqrt{2}$	$1.90 + \sqrt{2}$
a_3	2.85	0	$\sqrt{3}$	$2.85 + \sqrt{3}$
a_4	4.12	0	$\sqrt{4}$	$4.12 + \sqrt{4}$
a_5	1.90	0	$\sqrt{2}$	$1.90 + \sqrt{2}$
a_6	0.95	1	$\sqrt{1}$	$0.95 + \sqrt{1}$
a_7	1.27	1	$\sqrt{1}$	$1.27 + \sqrt{1}$
a_8	0.95	1	$\sqrt{1}$	$0.95 + \sqrt{1}$

- Scores of each link of the network of Figure 1 for $U = \emptyset$.
- The first selected link is a_1: $U = \{a_1\}$.

	$f_1(\{a_1\}, a_i)$	$f_2(\{a_1\}, a_i)$	$f_3(\{a_1\}, a_i)$	$W(\{a_1\}, a_i)$
a_2	0.00	0	$\sqrt{2}$	$\sqrt{2}$
a_3	0.00	1	$\sqrt{3}$	$1 + \sqrt{3}$
a_4	0.00	1	$2\sqrt{2}$	$1+2\sqrt{2}$
a_5	0.00	0	$\sqrt{2}$	$\sqrt{2}$
a_6	0.00	1	0	1
a_7	0.00	1	0	1
a_8	0.00	1	0	1

- The second selected link is a_4: $U = \{a_1, a_4\}$.

	$f_1(\{a_1, a_4\}, a_i)$	$f_2(\{a_1, a_4\}, a_i)$	$f_3(\{a_1, a_4\}, a_i)$	$W(\{a_1, a_4\}, a_i)$
a_2	0.00	5	0	5
a_3	0.00	5	0	5
a_5	0.00	1	$\sqrt{2}$	$1 + \sqrt{2}$
a_6	0.00	1	0	1
a_7	0.00	2	0	2
a_8	0.00	1	0	1

- The third selected link is a_2 (or also a_3): $U = \{a_1, a_4, a_2\}$.
- The current set U is feasible.
- The re-optimization step does not eliminate any element from U.
- The final returned solution is optimal.

Figure 3: Steps of Greedy2 algorithm when it is applied to solve problem P1 on the network of Figure 1.

FindNeighbor(U)

1. Let $U' \leftarrow U$;
2. Randomly select $a \in U'$;
3. **For** h iterations **do**
   3.1 Randomly select $R_i \in Cov(\emptyset, a)$;
   3.2 Randomly select, if it exists, a link $\hat{a}$ such that $\hat{a} \in U \cap R_i$;
   3.3 Set $U' \leftarrow U' \backslash \{\hat{a}\}$;
4. Add elements to U' by applying Greedy2 until a feasible solution $\hat{U}$ is obtained;
5. Return $\hat{U}$.

Table 9: *The procedure FindNeighbor(U).*

4.3 Tabu Search Approach

In this section we describe the Tabu Search approach we developed to solve problem P1. Given a feasible solution $U \subseteq A$ for the problem, we defined the neighborhood $N_h(U)$ of this solution as the set of all the feasible solutions obtained from U by deleting at most h elements. Formally, the following neighborhood is considered:

k-switch Neigborhood $N_h(U)$

A subset of links $U' \in N_h(U)$ if and only if U' is such that: $|U' \cap U| \geq |U| - h$, that is the two subsets U and U' share at least $|U| - h$ links.

Note that, a neighbor $U' \in N_h(U)$ can be obtained from U by randomly deleting h elements and adding new elements to restore feasibility; such a procedure, referred to as *FindNeighbor(U)*, is described in Table 9. The value of parameter h must be, on the one hand, large enough to ensure to find a neighbor $U' \in N(U)_h$ such that $U' \neq U$, and, on the other hand, its value should vary according to the size of the instance. After a tuning phase we set $h = 5 + \beta(|U|)$, where $\beta(|U|)$ is a random value between 0 and $\frac{|U|}{10}$. The tabu list memorizes moves corresponding to feasible solutions. Ideally, the tabu list should store the subset of links of each solution that is contained in the tabu list. Since memorizing such a detailed information could be heavy from a memory point of view, we decided to store in the tabu list an hash key $H(U)$ that is univocally associated with solution U. When the algorithm needs to verify whether a given solution is in the tabu list it looks for its hash key in the tabu list. To give an example, let us consider $U = \{a_{i_1}, a_{i_2}, \ldots, a_{i_s}\}$ to be a feasible solution composed of $|U|$ links, where a_{i_j} is the identification ID of the link. The corresponding hash key $H(U)$ is obtained by applying the following steps:

- Order the links in U in non decreasing order with respect to their ID and obtain: $a_{i_{j_1}} \leq a_{i_{j_2}} \leq , \ldots, \leq a_{i_{j_{|U|}}}$
- $\gamma(0) = 1$

Iteration i-th of Tabu Search

1. Apply *FindNeighbor(U)* and obtain $\hat{U}$;
2. If $|\hat{U}| < |U^*|$ then $U^* \leftarrow \hat{U}$, $T \leftarrow T \cup \{H(U^*)\}$;
3. If $|\hat{U}| \geq |U^*|$ and $H(\hat{U}) \notin T$ and $|\hat{U}| < |U^{**}|$ then $U^{**} \leftarrow \hat{U}$;
4. If after p iterations the incumbent solution U^* was never updated then $U^{**} \leftarrow U^*$;

Table 10: *The i-th iteration of our tabu search algorithm*

- $\gamma(h) = \alpha_1 a_{i_{j_h}} + \alpha_2 \gamma(h-1)$
- $H(U) = \gamma(|U|)$

The first step is to order the links in the solution with respect to their ID. This step ensures that the hash key associated with a given solution does not depend on the order of the links in the solution. The hash key associated with U is a value computed by means of a recursive function $\gamma(\cdot)$. The value $\gamma(h)$ is associated with the link in position h, $h = 1, 2, \ldots, |U|$, in the ordered set U; such a value is computed according to the value of the recursive function $\gamma(h-1)$ associated with the link in the previous position $h-1$. The hash key associated with U is the value of the recursive function computed for the link in the last position. The values α_1 and α_2 are prime numbers. If the links were identified by prime numbers the resulting hash key would be unique. Of course, due to the size of the instance of the problem such an identification for the links is not possible. This results in a key that may not be unique. To limit the cases of different feasible solutions associated with the same hash key value, one would choose the values of α_1 and α_2 as greater as possible. In our experimentation we set $\alpha_1 = 4523$ and $\alpha_2 = 31$.
Finally, we chose to implement a tabu list with a fixed length equal to 8. The generic iteration step of our tabu search algorithm is described in Table 10 where U is the current feasible solution, U^* is the current incumbent solution, U^{**} is the second incumbent solution and T is the current tabu list. The value of parameter p is fixed in our experiments equal to $|U| * 5$. The initial feasible solution to start the first iteration of our tabu search is obtained by applying Greedy2. Our Tabu Search terminates if no improvement in the incumbent solution is obtained after $|N| * 150$ iteration, where $|N|$ is the total number of nodes of the input graph.

4.4 Adaptation to solve problem P1b

Our solution approaches described to solve problem P1 can be easily adapted also to solve problem P1b and therefore to take into account the cost c_a of installation of a sensor associated with each link.
The modifications for Greedy1 involve the definition of the Initial Ranking and of the greedy step criterion.

The *Initial ranking* is such that the links are ordered increasingly with respect to the link cost; ties are broken according first to a decreasing order with respect to the Covering Criterion (computed for $U = \emptyset$) and then according to a decreasing order with respect to the Diversification Criterion (computed for $U = \emptyset$).

The greedy Step 2.1 of the Greedy1-P1 (in Table 4) becomes:

Step 2.1. While U is not a feasible solution

- find the link $a \in A \backslash U$ with $\frac{|Cov(U,a)|}{c_a}$ maximum;
- in case of ties select the one with $\frac{|Div(U,a)|}{c_a}$ maximum;
- in case of ties select the one with the best position in the *Initial Ranking*.

The modifications for Greedy2 involve the greedy criterion $W(U, a)$, that becomes $\frac{W(U,a)}{c_a}$. Such a modification is also considered in the Tabu Search algorithm. Moreover, Step 3 in the i-th iteration in the Tabu Search (in Table 10) is also changed to take into account the new objective function. It becomes:

3. If $c(\hat{U}) \geq c(U^*)$ and $H(\hat{U}) \notin T$ and $c(\hat{U}) < c(U^{**})$ then $U^{**} \leftarrow \hat{U}$.

where $c(U) = \sum_{a \in U} c_a$ is the sum of the costs of the links in the set U.

4.5 Adaptation to solve problem P2 and P2b

The adaptation of the approaches to solve problem P2 and P2b requires to take into account a new objective function (the maximization of total weight of the routes that are observable) and the added budget constraint involving either the total number of sensors that can be located (problem P2) or the total cost of the sensors that can be located (problem P2b). All the changes are described for problem P2b and reported below. The same changes can be applied to solve problem P2 by setting $C_{max} = k$ and $c_a = 1$, $\forall a \in A$. Let $z(U)$ define the total weight of the routes that can be univocally determined if vehicle-ID sensors are located on the links in U.

The modifications for Greedy1 algorithm involve the Initial Ranking and the Greedy criterion. When solving problem P2b the Initial Ranking takes into account the cost associated with each link and it is the same we considered for problem P1b, that is, the links are ordered increasingly with respect to the link cost; ties are broken according first to a decreasing order with respect to the Covering Criterion (computed for $U = \emptyset$) and then according to a decreasing order with respect to the Diversification Criterion (computed for $U = \emptyset$). The greedy criterion to select link a is modified to take into account both the total number of additional constraints that can be satisfied, and the weights of the additional covered routes. In particular, the algorithm selects at each iteration a link a such that: (i) the budget

Greedy1 - P2b

1. `Initialization`
   Let $U \leftarrow \emptyset$ be the set of selected links.
   Let A be the set of links in the network.
2. `Greedy step`
   2.1 While $(c(U) < C_{max})$ and $(A \backslash U \neq \emptyset)$
   find the link $a \in A \backslash U$ with $\frac{|Cov(U,a)|+|Div(U,a)|}{c(a)} + \frac{\sum_{R_i \in Cov(U,a)} w_{R_i}}{\bar{w}}$ maximum
   in case of ties select the link with the best position in the *Initial Ranking*;
   if $(c(U) + c(a)) \leq C_{max}$ then $A \leftarrow A \backslash \{a\}$ and $U \leftarrow U \cup \{a\}$;
   else $A \leftarrow A \backslash \{a\}$
3. Return U

Table 11: *Greedy1 Algorithm to solve Problem P2b.*

constraint is not violated and (ii) the following quantity is maximum

$$\frac{|Cov(U,a)| + |Div(U,a)|}{c(a)} + \frac{\sum_{R_i \in Cov(U,a)} w_{R_i}}{\bar{w}}$$

where the first element is the ratio between the sum of the covering and diversification scores over the cost of the link; while the second element is the ration between the sum of the weights of the additional covered routes divided by the average route weight $\bar{w} = \frac{\sum_{R_i \in R} w_{R_i}}{|R|}$. The pseudocode is given in Table 11. Modifications of the Greedy2 algorithm involves the stopping criterion and the values of the parameters ρ_i, $i = 1, 2, 3$ associated to the three components that are involved in the greedy criterion. The new values assigned to parameters ρ_i are: $\rho_1 = 100$, $\rho_2 = 2\rho_1$ and $\rho_3 = 1$; this setting favors the choice of those links that diversify most. Greedy2 stops when the given budget limit is reached. Finally, the same modifications apply to the Tabu Search algorithm. Moreover, Step 3 in the i-th iteration in the Tabu Search is also changed to take into account the new objective function. It becomes:

3. If $z(\hat{U}) \leq z(U^*)$ and $H(\hat{U}) \notin T$ and $z(\hat{U}) > z(U^{**})$ then $U^{**} \leftarrow \hat{U}$.

5 Experimentation

In this section we show the experimental results to test the performance of our approaches, in terms of solution quality and running time, and to evaluate when our proposed enhanced mathematical formulations provide a different optimal solutions than the formulations presented by Castillo et al. [9] and Minguez et al. [22].

Since there do not exist benchmark instances in the literature, we generated two different classes of instances. The first one, referred to as Class 1, is a set of instances where the set of routes do not show

the circular pattern behavior described in section 3. The second class of instances, referred to as Class 2, contains instances where the set of routes in the network show a circular behavior. These instances are useful to evaluate the effectiveness of our enhanced mathematical formulations in providing better solutions with respect to those provided by the existing mathematical formulations.

Class 1 and Class 2 instances are downloadable from the authors' website [1] On each instance, we compare the results returned by the mathematical models and by our Greedy 1, Greedy 2 and Tabu search algorithms for each of the four problems. The greedy algorithms and the Tabu Search algorithm were coded in C and run on a 3.4 Ghz Intel i7 processor. The mathematical formulations were coded in AMPL and solved by means of the solver CPLEX 11.2. We fixed a threshold on the execution time of the CPLEX solver equal to 2 hours.

5.1 Instances Description

Class 1 set of instances are modified square grid graphs, where arcs are added randomly between pairs of vertices of the graph. In particular, a number of arcs equal to 10% of the total number of arcs of a regular grid were randomly added to each grid. We generated square grids whose dimensions are: 8×8, 10×10 and 15×15, for a total number of nodes equal to $n = 64, 100$ and 225, respectively. For each grid we generated a fixed number $|R|$ of routes such that $|R| = n, 2n, 3n, 4n, 5n$. Hence, we generated 15 different scenarios. Costs on links were generated randomly from a uniform variable and they range from 1 to 5. Weights on routes were generated randomly from a uniform variable and they range from 10 to 100. For each scenario, 5 different instances were generated.

Class 2 set of instances are generated considering networks with a total number of nodes equal to $n = 40, 80, 160, 240, 320, 400$ and corresponding number of OD pairs equal to $\frac{n}{4}$. A fixed number $|R|$ of routes was generated for each OD pair with $|R| = 4, 8, 10, 16$. We then generated 24 different scenarios. Costs on links were generated randomly from a uniform variable and they range from 1 to 5. Weights on routes were generated randomly from a uniform variable and they range from 10 to 100. For each scenario, ten different instances were generated.

5.2 Results

All the results are showed in Tables 12-18. Since computational times and memory requirements to solve the mathematical models hugely increase with the size of the instances, we report the results of our models for the first 40 instances of Class 1 and the first 60 instances of Class 2, being the time limit reached for problem P2 and problem P1 for higher dimensional instances. The detailed results on each instance

on these limited set, returned by the mathematical models, can be downloaded from the authors' website [1]. Comparisons of the solutions returned by our heuristic approaches with the optimal solution (when available), on these limited set of instances, are reported in Table 13 and Table 14 Finally, comparisons of the quality of the solutions returned by our heuristic approaches on the entire set of instances are given in Tables 15 - 18.

To compare the effectiveness of the new set of diversification constraints Table 12 shows, for each problem and each class of instances, the percentage of instances where our mathematical formulation returned a better solution than the one returned by the models in the literature. Hence, for example, the percentage of instances of Class 2 where the solution of our mathematical formulation of problem P1 returned a different (and hence better) solution that the one returned by the existing mathematical formulation is equal to 80%, and on the remaining 20% of the instances the two mathematical formulations returned the same solution. The new set of diversification constraints reveals to be more effective in diversifying routes related to Class 2 instances than those of Class 1 instances. Note that, the budget constraints for problem P2b makes the new diversification constraints not effective at all.

The analysis of the results in the detailed tables downloadable from the website reveal the average gap between the optimum solution is 1% on Class 1 instances and 3% on the instances of Class 2. Our proposed models require more computational time to be solved than those existing in the literature. This was an expected result being the total number of variables and of constraints greater. Additionally, comparing the computational times of the instances in Class 1 and Class 2 with the same number of routes, we can observe that higher computational times are required to solve instances of Class 1 than those of Class 2, revealing that graphs where the routes show a circular pattern are more easy to deal with as it was expected. Among the four problems, problem P1b and problem P2b seem easier (in terms of required computational time) to be solved than problems P1 and P2. In particular, the time limit to solve our mathematical formulation of problem P1, P1b, P2 and P2b on Class 1 instances was reached, respectively, on 10, 0, 33, 1 instances out of 40. Moreover, our mathematical formulation was not able to find a feasible solution within the given time limit for Problem P2 on two instances (namely, instance 37 and instance 39) of Class 1. While, the time limit to solve our mathematical formulation of the four problems was never reached on the 60 considered instances of Class 2 when solving problems P1, P1b, and P2b, while it was reached on 8 instances out of 60 when solving problem P2. This can be explained with the fact that having different cost associated with the arcs allows, on the one hand, to obtain better bounds that allow to cut off part of the branch and bound tree (for problem P1b), while, on the other hand, to prune the tree tanks to the budget constraint (for problem P2b).

Tables 13 and 14 compare the quality of the solutions provided by Greedy1, Greedy2 and the Tabu Search algorithm with the optimum solution (when available) on the 40 instances of Class 1 and the 60

	P1	P1b	P2	P2b
Class 1	20%	18%	50%	0%
Class 2	80%	82%	95%	0%

Table 12: Comparison of the mathematical formulations: for each problem and each class of instances, the percentage of instances where our mathematical formulation returned a better solution than the one returned by the models in the literature is given.

instances of Class2 on which we run the mathematical formulations to optimality. In each table, the first three columns report the characteristics of each scenario: scenario ID, the number of nodes (n) and the number of routes (r). For each algorithm, the tables report the average values taken on the tested instances of the relative gap between the returned solution and the optimum solution, when available. In particular, this gap is computed as the ratio $\frac{UB-Optimum}{UB}$ for problem P1 and P1b, and $\frac{Optimum-LB}{LB}$ for problem P2 and P2b, where UB and LB are the value of the bounds returned by our algorithms. Whenever at least one instance of the scenario has not been solved to optimality by the solver CPLEX within the time limit of 7200 secs., the term NA (Not Available) appears in the tables. In such case, of course, no gap value is reported for all the tested algorithms.

Tables 15 - 18 compare the solutions and the computational time of Greedy1, Greedy2 and the Tabu Search algorithm on all the instances of Class1 and Class2. In each table, the first three columns report the characteristics of each scenario: scenario ID, the number of nodes (n) and the number of routes (r). For each algorithm, the tables report the average values for each scenario taken on the tested instances (5 instances for Class1 and 10 Instances for Class 2) of the bound and of the computational time (in seconds)

			P1			P1b			P2			P2b		
Scenario	**n**	**r**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**
1	64	64	22%	17%	1%	5%	9%	2%	24%	25%	2%	35%	28%	25%
2	64	128	23%	10%	0%	7%	9%	0%	NA	NA	NA	53%	52%	47%
3	64	192	22%	13%	1%	8%	10%	0%	NA	NA	NA	65%	60%	53%
4	64	256	16%	17%	0%	10%	10%	1%	NA	NA	NA	68%	66%	61%
5	64	320	NA	NA	NA	9%	11%	0%	NA	NA	NA	NA	NA	NA
6	100	100	24%	18%	0%	11%	11%	1%	NA	NA	NA	21%	15%	8%
7	100	200	NA	NA	NA	9%	17%	1%	NA	NA	NA	40%	36%	28%
8	100	300	NA	NA	NA	11%	10%	1%	NA	NA	NA	45%	42%	37%

Table 13: Percentage gap between the solution returned by our algorithms and the optimum solution (when available) on the instances of Class 1.

			P1			P1b			P2			P2b		
Scenario	**n**	**r**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**	**Greedy 1**	**Greedy 2**	**Tabu**
1	40	40	8%	4%	0%	4%	1%	0%	19%	12%	6%	21%	15%	12%
2	40	80	10%	8%	1%	6%	3%	1%	14%	10%	4%	21%	15%	10%
3	40	100	10%	9%	2%	5%	5%	1%	10%	8%	4%	21%	19%	12%
4	40	160	12%	9%	2%	8%	4%	2%	NA	NA	NA	22%	20%	15%
5	80	80	3%	2%	0%	2%	1%	0%	31%	8%	4%	21%	12%	10%
6	80	160	5%	4%	0%	3%	2%	0%	14%	9%	4%	22%	16%	11%

Table 14: Percentage gap between the solution returned by our algorithms and the optimum solution (when available) on the instances of Class 2.

The percentage gap from the optimum of the tabu search is always better than those showed by Greedy1 and Greedy2 as it is expected. This better performance is, however, paid in terms of greater computational time as it is evident by the analysis of tables 15 - 18. The percentage gap of the Tabu Search ranges between 0% and 2% when solving problems P1 and P1b. These gaps are higher when solving problem P2 and P2b. In particular, the gaps for problem P2 are not available for many of the instances of Class 1 (table 13) while the gaps for problem P2 on instances of Class 2 varies between 4% and 6% (table 14). Finally, the tabu search, when solving problem P2b has a gap that ranges from 8% to 61% on Class 1 instances and a gap that ranges from 10 % to 15 % on Class 2 instances. The solution returned by Greedy2 algorithm is consistently better than that returned by Greedy1. The two algorithms are however comparable in terms of computational times which are always less than one second, but for Greedy1 algorithm on scenarios 20, 23 and 24 on instances of Class 2 for problem P2 and on scenarios 16,20,22, 23 and 24 on instances of Class 2 for problem P2b (table 18).

			P1						P1b					
Scenario	**nn**	**rr**	**Greedy 1**		**Greedy 2**		**Tabu**		**Greedy 1**		**Greedy 2**		**Tabu**	
			Sol	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**
1	64	64	19,20	0,000	18,00	0,000	15,20	1,734	33,40	0,000	34,80	0,000	32,20	2,472
2	64	128	27,20	0,000	23,20	0,000	20,80	4,166	48,60	0,000	50,00	0,000	45,40	8,242
3	64	192	32,60	0,000	29,40	0,000	25,60	6,736	61,60	0,004	62,60	0,004	56,80	13,884
4	64	256	34,40	0,002	34,60	0,008	28,80	9,146	73,80	0,010	74,00	0,000	66,80	17,000
5	64	320	39,60	0,010	36,40	0,000	32,00	13,520	79,00	0,016	81,00	0,002	72,00	28,504
6	100	100	27,00	0,000	24,80	0,000	20,40	5,062	38,40	0,000	38,40	0,000	34,20	8,602
7	100	200	36,60	0,000	37,20	0,010	30,00	11,962	62,80	0,008	69,00	0,002	57,40	22,876
8	100	300	43,40	0,010	43,00	0,000	35,80	20,440	84,80	0,020	84,20	0,004	76,00	35,510
9	100	400	50,20	0,020	48,80	0,002	40,80	28,430	93,60	0,036	99,20	0,010	89,60	55,112
10	100	500	54,60	0,028	54,00	0,006	44,60	42,192	105,80	0,052	111,60	0,014	99,00	86,436
11	225	225	51,40	0,010	49,40	0,010	40,20	35,670	77,20	0,020	76,00	0,012	69,20	64,422
12	225	450	68,60	0,036	68,40	0,016	55,60	97,222	112,00	0,070	111,80	0,024	101,40	200,838
13	225	675	83,20	0,132	82,00	0,030	66,80	226,590	144,40	0,180	147,60	0,044	131,80	289,896
14	225	900	94,80	0,286	92,60	0,048	76,80	339,322	161,20	0,330	164,20	0,066	147,80	527,072
15	225	1125	103,60	0,482	103,80	0,078	84,80	624,202	194,80	0,436	205,00	0,078	180,40	546,658

Table 15: Comparison of the algorithms on instances of Class 1 for problems P1 and P1b.

			P2						P2b					
Scenario	**nn**	**rr**	**Greedy 1**		**Greedy 2**		**Tabu**		**Greedy 1**		**Greedy 2**		**Tabu**	
			Sol	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**
1	64	64	2305,60	0,000	2267,40	0,000	2950,80	2,884	2299,60	0,000	2522,00	0,000	2657,40	3,802
2	64	128	2819,60	0,000	2920,40	0,000	3840,40	6,904	3190,80	0,000	3273,00	0,000	3601,00	8,462
3	64	192	3029,80	0,000	3457,20	0,000	4713,00	11,916	3683,40	0,000	4137,20	0,002	4956,60	14,668
4	64	256	3488,80	0,002	4216,20	0,006	5316,20	15,494	4505,20	0,004	4753,40	0,004	5506,80	19,782
5	64	320	3912,40	0,008	3876,00	0,000	5873,60	19,856	5791,00	0,010	6408,60	0,000	7114,00	25,804
6	100	100	4544,40	0,000	4707,80	0,000	5485,80	9,666	4440,60	0,000	4746,00	0,000	5148,80	10,490
7	100	200	5727,40	0,008	6702,20	0,002	8317,80	21,598	6685,40	0,010	7117,20	0,000	7948,60	29,662
8	100	300	7232,60	0,010	7725,00	0,004	10263,60	34,516	8916,40	0,014	9401,20	0,004	10256,80	54,838
9	100	400	7747,20	0,022	9028,80	0,008	11585,80	66,348	9749,00	0,024	10454,80	0,008	11669,60	66,586
10	100	500	8884,20	0,036	10231,40	0,006	13075,60	90,314	11029,40	0,042	11689,60	0,006	12953,40	124,318
11	225	225	11708,40	0,020	12001,40	0,008	12378,20	44,624	11670,00	0,022	11650,00	0,010	12268,40	47,522
12	225	450	19694,80	0,052	20569,80	0,018	23027,80	223,122	20040,00	0,068	20682,20	0,020	22054,60	178,098
13	225	675	25887,00	0,146	27703,80	0,028	31499,60	406,208	27634,00	0,168	28807,00	0,036	30875,40	367,940
14	225	900	30244,40	0,258	33665,00	0,038	38327,60	576,708	34931,00	0,304	36014,40	0,046	38553,40	430,276
15	225	1125	34177,20	0,322	38020,60	0,038	44187,00	714,928	39961,40	0,414	41257,20	0,060	44091,80	516,958

Table 16: Comparison of the algorithms on instances of Class 1 for problems P2 and P2b.

			P1						P1b					
Scenario	**nn**	**rr**	**Greedy 1**		**Greedy 2**		**Tabu**		**Greedy 1**		**Greedy 2**		**Tabu**	
			Sol	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**
1	40	40	29,80	0,000	28,60	0,000	27,60	0,172	45,10	0,000	44,00	0,000	43,50	0,210
2	40	80	52,50	0,000	51,30	0,000	48,00	0,472	85,40	0,000	83,40	0,000	81,50	0,499
3	40	100	62,40	0,000	61,30	0,000	57,00	0,665	106,50	0,000	105,80	0,002	101,70	0,684
4	40	160	88,70	0,006	85,30	0,004	79,10	1,711	161,90	0,010	155,40	0,000	151,40	1,303
5	80	80	68,10	0,000	67,40	0,000	66,40	0,649	97,80	0,000	96,20	0,000	95,60	0,710
6	80	160	120,00	0,010	118,50	0,001	113,80	1,958	190,00	0,010	187,90	0,009	184,20	1,852
7	80	200	143,80	0,010	141,10	0,007	135,30	2,627	234,40	0,015	229,20	0,013	225,00	2,886
8	80	320	207,50	0,024	202,60	0,018	190,00	7,167	342,80	0,028	336,00	0,026	328,80	5,121
9	160	160	145,10	0,010	144,20	0,008	143,20	2,400	197,60	0,010	194,00	0,009	193,30	2,927
10	160	320	267,10	0,027	263,00	0,026	258,90	6,652	393,50	0,029	388,10	0,034	384,50	7,441
11	160	400	321,20	0,041	315,30	0,040	308,30	11,500	479,70	0,052	473,10	0,051	466,50	9,827
12	160	640	472,00	0,106	464,80	0,090	448,70	29,607	737,00	0,121	723,20	0,133	712,30	21,615
13	240	240	223,90	0,021	223,30	0,019	222,30	6,238	296,20	0,025	292,30	0,019	291,60	7,324
14	240	480	419,70	0,071	416,50	0,065	412,80	13,762	591,80	0,086	585,10	0,083	580,30	18,139
15	240	600	512,40	0,108	508,20	0,105	501,70	21,631	740,00	0,125	726,30	0,131	720,90	24,547
16	240	960	765,60	0,285	756,60	0,269	740,30	68,186	1145,00	0,349	1127,50	0,382	1112,30	61,273
17	320	320	304,60	0,036	303,70	0,036	302,40	11,515	399,90	0,039	395,20	0,042	393,50	15,025
18	320	640	576,80	0,142	572,70	0,137	567,90	30,915	783,70	0,155	772,30	0,165	767,70	42,082
19	320	800	701,30	0,215	695,90	0,207	688,60	50,282	980,40	0,237	965,50	0,269	958,80	54,555
20	320	1280	1057,70	0,562	1046,80	0,520	1030,30	109,224	1554,00	0,659	1536,20	0,732	1518,10	143,707
21	400	400	380,90	0,055	380,50	0,056	379,70	17,365	503,30	0,058	496,00	0,063	494,40	25,308
22	400	800	731,40	0,233	728,30	0,231	724,30	48,943	1002,20	0,256	991,20	0,284	986,40	66,792
23	400	1000	899,30	0,356	894,70	0,352	888,80	72,718	1239,60	0,382	1223,00	0,432	1215,90	95,081
24	400	1600	1363,40	0,919	1354,60	0,865	1335,90	174,387	1963,60	0,927	1939,80	1,033	1920,00	249,132

Table 17: Comparison of the algorithms on instances of Class 2 for problems P1 and P1b.

			P2						P2b					
Scenario	**nn**	**rr**	**Greedy 1**		**Greedy 2**		**Tabu**		**Greedy 1**		**Greedy 2**		**Tabu**	
			Sol	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**	**Sol**	**Time**
1	40	40	1612,10	0,000	1755,60	0,000	1865,20	0,334	1695,10	0,000	1807,20	0,000	1884,10	0,191
2	40	80	3783,40	0,000	3964,20	0,000	4194,00	1,292	3571,20	0,000	3816,90	0,000	4045,80	0,475
3	40	100	4914,10	0,000	5013,60	0,000	5252,50	1,719	4444,50	0,000	4554,50	0,000	4929,40	0,672
4	40	160	7819,70	0,010	8012,80	0,000	8427,60	4,250	6878,90	0,000	7045,00	0,009	7525,60	1,551
5	80	80	2730,30	0,000	3635,60	0,000	3781,30	1,883	3470,50	0,000	3866,50	0,000	3982,80	0,931
6	80	160	7165,00	0,010	7601,70	0,009	8015,50	7,015	6888,80	0,010	7407,80	0,000	7853,10	2,960
7	80	200	9062,10	0,020	9525,00	0,010	10066,70	12,199	8604,00	0,011	9339,80	0,008	9883,30	5,255
8	80	320	15408,20	0,046	15838,30	0,021	16754,90	29,257	14215,90	0,037	15050,70	0,019	15886,50	11,560
9	160	160	4765,50	0,016	6995,60	0,002	7361,20	12,510	6801,40	0,010	7650,70	0,002	7910,80	6,295
10	160	320	11867,30	0,066	14427,40	0,025	15377,00	57,212	13766,10	0,041	14917,60	0,020	15604,40	22,948
11	160	400	15509,20	0,118	18004,80	0,042	19350,60	95,534	17409,10	0,079	18782,40	0,030	19686,60	33,792
12	160	640	27568,90	0,402	29683,10	0,102	31830,10	227,508	28105,90	0,268	30014,40	0,080	31569,90	108,472
13	240	240	6540,80	0,041	10662,40	0,017	11045,10	33,888	10193,60	0,030	11423,70	0,010	11744,80	18,937
14	240	480	15818,90	0,219	21170,80	0,058	22445,90	142,570	20451,40	0,140	22571,00	0,048	23460,80	90,943
15	240	600	20704,10	0,330	26470,90	0,077	28128,80	165,310	25445,20	0,271	27999,40	0,075	29155,60	141,094
16	240	960	37701,30	0,802	43767,70	0,155	46710,70	575,173	41685,80	1,028	45376,00	0,194	47359,30	368,709
17	320	320	8327,60	0,041	14005,40	0,017	14402,70	46,142	13388,00	0,049	15028,70	0,021	15507,00	43,605
18	320	640	19762,90	0,305	28345,70	0,069	29818,30	232,582	27650,30	0,342	30583,80	0,090	31553,40	205,412
19	320	800	26671,40	0,560	35367,50	0,108	37418,70	439,450	33993,30	0,662	37762,10	0,138	39246,20	299,741
20	320	1280	48069,20	2,031	57222,20	0,286	61342,50	1531,164	54811,90	3,187	60088,30	0,422	62635,40	1076,218
21	400	400	10237,40	0,087	17569,90	0,028	17998,80	83,878	16398,60	0,167	18473,20	0,055	19189,20	136,849
22	400	800	23297,90	0,605	35204,30	0,111	36906,20	422,229	33624,90	1,274	37532,80	0,234	38925,80	568,778
23	400	1000	30226,90	1,110	43934,80	0,171	46174,70	624,979	42081,30	1,930	46793,10	0,281	48635,10	732,775
24	400	1600	55948,50	4,733	70725,60	0,497	75501,60	2286,816	68490,40	5,180	74928,90	0,553	77957,90	1392,026

Table 18: Comparison of the algorithms on instances of Class 2 for problems P2 and P2b.

6 Conclusions

We addressed an important problem in the context of traffic control and management related to the optimum location of vehicle-ID sensors on the links of a network to derive route flow volumes. We considered both the full observability version of the problem where one seeks for the minimum number of sensors (or minimum cost) such that all the route flow volumes can be derived and the estimation version of the problem that arises when there is a limited budget in the location of sensors. Our contribution is both theoretical and algorithmic. We improved the existing mathematical formulations of the four addressed problems by better describing the feasible region. Being the problems NP-complete, we proposed three solution approaches (still missing in the literature) to heuristically solve the problems: two greedy algorithms and a tabu search metaheuristic. We tested our approaches on two different set of instances to better evaluate the quality of solutions provided by our algorithms and by the mathematical models when the set of routes show a circular pattern behavior or not.

The proposed analysis is developed under the assumption (common to other papers existing in the literature [9], [22]) that the information regarding OD routes is correct and complete. An interesting development of this research is now focused on the analysis of the sensitivity of the solutions with respect to the level of knowledge of the OD routes.

References

[1] *http://www.dmi.unisa.it/people/gentili/www/PublicationM.htm.*

[2] E. Castillo andI. Gallego, J.M. Menéndez, and A. Rivas. Optimal use of plate-scanning resources for route flow estimation in traffic networks. *IEEE Transaction on Intelligent Transportation Systems*, 11:380–391, 2010.

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

[4] E. Castillo, A.J. Conejo, J.M. Menéndez, and P. Jiménez. The observability problem in traffic network models. *Computer-Aided Civil and Infrastructure Engineering*, 23:208–222, 2008.

[5] E. Castillo, A.J. Conejo, R.E. Pruneda, and C. Solares. Observability in linear systems of equations and inequalities: Applications. *Computers and Operations Research*, 34:1708–1720, 2007.

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

[7] E. Castillo, M. Nogal, A. Rivas, and S. Snchez-Cambronero. Observability of traffic networks. optimal location of counting and scanning devices. *Transportmetrica B: Transport Dynamics*, 1(1):68–102, 2013.

[8] P. Chootinan, A. Chen, and H. Yang. A bi-objective traffic location problem for origin -destination trip table estimation. *Transportmetrica*, 1:65–80, 2005.

[9] E.Castillo, J.M. Menéndez, and P. Jiménez. Trip matrix and path flow reconstruction and estimation based on plate scanning and link observations. *Transportation Research Part B*, 42, 2008.

[10] Stacy M. Eisenman, Xiang Fei, Xuesong Zhou, and Hani S. Mahmassani. Number and location of sensors for real-time network traffic estimation and prediction: Sensitivity analysis. *Transportation Research Record*, 1964:253–259, 2006.

[11] A. Elhert, M.G.H. Bell, and S. Grosso. The optimization of traffic count locations in road networks. *Transportation Research Part B*, 40:460–479, 2006.

[12] X. Fei and H.S. Mahmassani. Structural analysis of near-optimal sensor locations for a stochastic large-scale network. *Transportation Research Part C*, 19:440–453, 2011.

[13] Xiang Fei, Hani S. Mahmassani, and Pamela Murray-Tuite. Vehicular network sensor placement optimization under uncertainty. *Transportation Research Part C: Emerging Technologies*, 29(0):14 – 31, 2013.

[14] M. Gentili and P. Mirchandani. Survey of models to locate sensors to estimate traffic flows. *Transportation Research Record*, 2243:108–116, 2011.

[15] M. Gentili and P. Mirchandani. Locating sensors on traffic networks: Models, challenges and research opportunities. *Transportation Research Part C*, 24:227–255, 2012.

[16] M. Gentili and P.B. Mirchandani. Location of active sensors on traffic network. *Annals of Operations Research*, 136:229–257, 2005.

[17] Shou-Ren Hu, Srinivas Peeta, and Chun-Hsiao Chu. Identification of vehicle sensor locations for link-based network traffic applications. *Transportation Research Part B: Methodological*, 43(89):873 – 894, 2009.

[18] H.Yang, C. Yang, and L. Gan. Models and algorithms for the screen line-based traffic counting location problem. *Computers and Operations Research*, 33:836–858, 2006.

[19] H.J. Kim, H. Chung, and S.Y. Chung. Selection of the optimal traffic counting locations for estimating origin-destination trip matrix. *Journal of Eastern Asian Society for Transportation Studies*, 5:1353 – 1365, 2003.

[20] W.H.K. Lam and H.P. Lo. Accuracy of o-d estimates from traffic counts. *Traffic Engineering and Control*, 31:435–447, 1990.

[21] X. Li and Y. Ouyang. Reliable sensor deployment for network traffic surveillance. *Transportation Research Part B*, 45:218–231, 2011.

[22] R. Mínguez, S. Sánchez-Cambronero, E. Castillo, and P. Jiménez. Optimal traffic plate scanning location for od trip matrix and route estimation in road networks. *Transportation Research Part B*, 44:282–298, 2010.

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

[24] Fulvio Simonelli, Vittorio Marzano, Andrea Papola, and Iolanda Vitiello. A network sensor location procedure accounting for od matrix estimate variability. *Transportation Research Part B: Methodological*, 46(10):1624 – 1638, 2012.

[25] Sheng xue He. A graphical approach to identify sensor locations for link flow inference. *Transportation Research Part B: Methodological*, 51(0):65 – 76, 2013.

[26] H. Yang and J. Zhou. Optimal traffic counting locations for origin-destination matrix estimation. *Transportation Research Part B*, 32:109–126, 1998.

[27] X. Zhou and G.F. List. An information-theoretic sensor location model for traffic origin-destination demand estimation applications. *Transportation Science*, 44:254–273, 2010.