Exact and approximate route set generation for resilient partial observability in sensor location problems

Marco Rinaldi, Ph.D. (corresponding author)
Research Associate
University of Luxembourg
Research Unit of Engineering Science,
Faculty of Science, Technology and Communication
6, Rue R. Coudenhove-Kalergi, L-1359 Luxembourg (Luxembourg)
Phone: +352 4666446281
marco.rinaldi@uni.lu

Francesco Viti, Ph.D.
Associate Professor
University of Luxembourg
Research Unit of Engineering Science,
Faculty of Science, Technology and Communication
6, Rue R. Coudenhove-Kalergi, L-1359 Luxembourg (Luxembourg)
Phone: +352 4666445352
francesco.viti@uni.lu

Abstract

Sensor positioning is a fundamental problem in transportation networks, as the location of sensors strongly determines how traffic flows are observable and hence manageable.

This paper aims to develop a methodology to determine sensor locations on a network such that an optimal trade-off solution is found between the amount of sensors installed and the resilience of the sensor set.

In particular, we propose exact and heuristic solutions for identifying the optimal route sets such that no other route would include any additional information for finding optimal full and partial observability solutions. This is an important contribution to sensor location problems, as route-based link flow inference problems have non-unique solutions, strongly depending on the used link-route information.

The properties of the new methodology are analysed and illustrated through different case studies, and the advantages of the algorithms are quantified both for full and for partial observability solutions. Thanks to the route sets found by our approach, we are able to find full observability solutions characterized by a small number of sensors, while yet being efficient also in terms of partial observability. We perform validation tests on both small and real-life sized network instances.

Keywords: *Network Sensor Location Problem, Partial Link Flow Observability, Route Set Generation, Maximum Clique Problem, Genetic algorithm*

1. Introduction

Traffic information and management applications strongly rely on how traffic flows are monitored. Locating traffic sensors in a network is therefore considered a problem of paramount importance in transportation engineering, in particular within estimation problems (e.g. real time traffic state estimation (Ahmed et al., 2014; Zhu et al., 2014), OD flows estimation (Hadavi and Shafahi, 2016; Hu and Liou, 2014; Zhou and List, 2010), link flow inference (Castillo et al., 2008c; Hu et al., 2009; Liu et al., 2014; Xu et al., 2016), travel time estimation (Viti et al., 2008; Xing et al., 2013) and path flow reconstruction (Cerrone et al., 2015; Fu et al., 2016, 2017; Li and Ouyang, 2011).

Among others, network flow observability is a class of the so-called Network Sensor Location Problems (NSLP), in which the main goal is to determine the minimum set of observed link, route or OD flows that can be measured to provide information on the remaining non-observed link (or route, or OD) flows. Solutions to these problems are approached in literature by exploiting the fundamental relationships between the three sets of variables (link flows, route flows, OD flows), derived from conservation of vehicles principles. In this study we specifically focus on the *link flow inference* problem, i.e. the problem of identifying a (smallest) set of independent links able to fully determine the flow on other links in the network (Castillo et al., 2015).

Existing approaches that compute link flow inference solutions are subdivided in methods exploiting node-link relations (Ng, 2012) or link-route relations (Castillo et al., 2008a; Hu et al., 2009). This latter category of approaches has been shown to potentially identify more efficient solutions. The work of (Viti et al., 2014) showed that, for mid- and large-sized networks, node-based approaches tend to recommend a systematically higher number of sensors to install if compared to their route-based counterparts. A fundamental reason for this systematic difference was observed in the richer information offered by link-route relations if compared to link-node relations.

Despite their theoretical relevance, full observability solutions are however of little practicality in real-life networks, as the number of links to equip with sensors easily grows beyond economic feasibility with the size and complexity of the network itself. In (Castillo et al., 2014) the authors reported that such solutions usually require around 60% of link flows to be observed, which is clearly not feasible in networks with hundreds or thousands of links. Given this argument, the concept of *partial* observability becomes very appealing. In partial observability problems, approaches must take into account different constraints (e.g. maximum number of sensors to be installed, location-specific restrictions, monetary budget limits,...), and therefore establish a trade-off between amount of measured/observed information and solution feasibility. The focus of the underlying problem shifts then from finding efficient measured link sets for complete link flow inference to, instead, maximizing the amount of *total* available information (different definitions of available information are provided later in Section 2 of this article). In (Viti et al., 2014) we developed a partial observability metric to characterize and classify partial observability solutions, showing how this metric was capable to capture partial relationships and locate sensors in a very intuitive manner. While exploring the properties of this metric we quickly realized that algebraically equivalent full observability solutions actually exhibit strong behavioural differences when analysed from a partial observability perspective.

More recently, in (Rinaldi et al., 2015) we refined these results by empirically assessing the influence that route enumeration criteria have on the shape and structure of the full observability solution. We specifically concentrated on studying how the amount and quality of information in either full or partial information solutions depends on the number of routes and on the composition of the route set. The main finding of the paper was that the overall information content tends to increase as the route set is expanded, but the rate of growth reduces non linearly with the route set size, suggesting

that there is an upper bound above which no new information is gained by including additional routes in the route set. Moreover, enumerating routes according to algebraic independence principles resulted in overall better-informative observability solutions with respect to standard enumeration techniques, a conclusion entirely in line with the findings of (Castillo et al., 2014).

However, due to the combinatorial nature of the problem at hand, in our previous approaches we have been limited to showcasing distributions of information contents, while incapable of isolating the single best possible combination of route set and, thus, independent/dependent variable sets.

In this work we attempt to bridge this methodological gap, by indeed studying the nature of information in observability problems. Throughout the rest of the paper we will develop methodologies to exactly and approximately generate such a route set, and empirically verify whether indeed the resulting full observability solution is optimal in terms of partial observability information content.

The remainder of this work is structured as follows: in Section 2 we present a concise literature review related to Network Sensor Location Problems and, more specifically, to observability problems. In Section 3 we introduce an exact methodology for determining the maximum independent route set for any given network as well as a heuristic algorithm, devised to overcome some of the exact algorithm's computational limitations. Section 4 details two test cases, based on several small to mid-sized networks, aimed at verifying the key hypothesis introduced in Section 3. In Section 5 a meta-heuristic approach is developed, in order to extend the results and insights obtained for smaller networks towards real life instances. Section 6 presents validation results of the metaheuristic approach wrt. the exact results obtained in Section 4, and, finally, results on two mid-large sized networks. Finally, in Section 7, conclusions and remarks related to future research are discussed.

2. Literature review

Network Sensor Location Problems

The literature related to NSLP can be separated into approaches focusing on the algebraic and topologic properties of the network structure and connections (*observability problems*) and in those relating observed traffic states (usually, flows) with the ones derived using estimation techniques (*flow-estimation problems*) or including behavioural models and/or statistical models to predict future traffic states (*flow prediction problems*). For a more extensive overview of these problem types, their formalisation and the relevant literature we refer to (Gentili and Mirchandani, 2012), (Viti et al., 2014) and (Castillo et al., 2015).

In observability problems, observed states from measurements can be used to infer this information to unobserved links states thanks to basic conservation of vehicles principles, i.e. when an unobserved variable can be related to only observed variables, it also becomes observable. In particular, in full observability solutions *all* unobserved flows are indirectly observable, as they all can be described as (linear) functions of the measured states. Flow-estimation problems, instead, seek to minimize the difference between measured and estimated traffic flows by applying various estimation techniques and/or heuristic rules (e.g., (Yang and Zhou, 1998), (Larsson et al., 2010), (Cipriani et al., 2006) and (Yang et al., 2006)). In this paper we focus on the first type, and in particular on the *link* flow observability problems or *link flow inference* problems, although the findings of the study are equally relevant for the second type, since most of the flow estimation problems rely on the same topological relations.

Link Inference problems

When dealing with link flow inference problems, most of the literature focuses on finding efficient *full* observability solutions, i.e. how to determine the minimum set of observed link flows such that complete link flow information can be determined through topological relationships alone. As mentioned in the introduction, these approaches usually exploit linear algebraic transformation techniques on incidence matrices to identify the set of observed flows, based e.g. upon node-link topological relationships (e.g., (Ng, 2012)), or upon link-route relationships (e.g., (Castillo et al., 2008a; Hu et al., 2009; Liu et al., 2014)). In both cases, the resulting solutions exhibit the very desirable property of guaranteeing linear independence relationships between all observed links, and exact linear dependences between unobserved and observed link flows. While the first relations are elegant as they do not require explicit route enumeration and remain tractable for large-sized networks, they do not contain all the information at the route and OD levels. This reflects in full observability solutions requiring a higher number of sensors to be installed, as empirically shown in (Viti et al., 2014). For this reason, our research follows a route-based approach.

In route-based link flow inference problems relationships between links are derived starting from the fundamental relations between link and route flows (see e.g. (Cascetta, 2009)):

$$v_l = \sum_{r \in R} a_{lr} h_r \forall l \in L \tag{1}$$

with h_r the route flow on a route r belonging to the routes set R, and v_l the link flow on link l belonging to the link set L. In this study we focus on static relations, which implies that the elements a_{lr} composing the link-route adjacency matrix can only have values of 0 or 1, depending on whether the given route r contains or not link l. The relations expressed in Equation (1) can also be formulated in vectorial-matrix form as:

$$v = Ah \tag{2}$$

Equations (1-2) express all linear relations connecting each state variable with other states in the network, provided that the route set R is exhaustive. A similar relation can be formulated between route and OD flows but given the focus of this paper on route set generation we restrict ourselves to relations (1-2).

One approach to solve link flow inference problems is to perform opportune matrix manipulations, i.e. by swapping or sorting rows or columns of A, while carefully maintaining the original relations between the variables as in the original set of relations (1-2). This approach (which we will refer to as *pivoting* procedure in this paper) has been proposed originally in (Castillo et al., 2001) for power networks, and later it has been extended to transportation networks (Castillo et al., 2008b, 2008a, 2011). One of the main advantages of the pivoting procedure is that, in any solution, the selected links are by construction linearly independent. Clearly, a disadvantage is that, given the combinatorial nature of the problem, multiple solutions usually exist, depending on the sequence in which rows and columns are manipulated, as well as how A is defined. Route-based approaches cannot guarantee in fact that all information about network topology and connectivity is included, if exhaustive route enumeration is not possible, which is very likely even in relatively small networks. This may then yield to solutions having different characteristics (or, ranks), i.e. full observability can be found with a different number (and position) of sensors. Hence, a minimum number of links to be observed is likely to exist, but is not guaranteed to be found through pivoting.

An alternative approach to the pivoting procedure has been proposed by (Hu et al., 2009), which studied the conversion of the matrix A into its "reduced row echelon form" through the Gaussian elimination method. By performing sensitivity analysis using different toy networks the authors suggested that an upper bound for the number of linearly independent links to be observed is likely

to exist, i.e. even if that number is variable, there must be an upper limit, above which any extra link flow will likely be linearly dependent on the others. In (Ng, 2012) the author showed that if a node-based approach is adopted, an analytical expression for this upper bound could be found: it equals the difference between the number of links *m* and the number of non-centroid nodes *n*. However, He (He, 2013) recently pointed out that the relation found by Ng does not consider all information contained in a network, as it neglects dependency of routes through the OD relations.

A more general goal is to find the *tight* upper bound of the minimum number of observed flows such that the system becomes fully observable, while, at the same time, the observed flows are all linearly independent, i.e. no redundant information is included. In (Castillo et al., 2014) the authors found that only linearly independent paths need to be used to obtain the minimum number of observed links. Through several numerical tests, they could show to improve the solutions provided by node-based approaches of up to (indicatively) 16% less sensors. In addition, they provided a graphical approach and some simple methods to obtain the maximum number of linearly independent paths and the maximum set of links to be counted in non-planar networks.

Partial Observability problems

Full observability solutions may require an exceedingly large amount of sensors to be placed, which is infeasible for any real-sized network. Hence, despite their neat theoretical meaning, solutions of full observability are rather impractical if not in very local problems (e.g. dealing only with few links and nodes).

Recently, the concept of *partial observability* has been introduced by different authors to identify solutions where full observability is not realistic. Partial observability solutions have been defined to answer two main research questions, i.e. 1) given some budget constraint (e.g. limited number of sensors to be installed) where sensors should be installed to maximize the information on the whole network? And 2) if one or more sensors are removed/added to an existing set, what is the information loss/gain on the whole network?

Gentili and Mirchandani (2012) were perhaps the first to provide a formal definition of partial observability solutions as the (minimum) subsets of link flow variables such that the system is partially observable at level *h*. In this work, *h* is defined as the sum of the observed states and the number of unmeasured variables, which are observable on the basis of the observed states. This definition was also used in e.g. (Castillo et al., 2011) and (He, 2013) to propose partial observability solutions under budget constraints. In (Viti et al., 2014) we discussed this definition and pointed out that, despite the rigorousness of the definition by (Gentili and Mirchandani, 2012), this does not provide any indication on the additional effort required to observe the unmeasured variables. In other words, since many partial observability solutions can be found with a specific level *h*, some of them may increase the value of *h* with adding one or in general fewer links than others. This can be desirable if one considers that new sensors could be installed later. A second argument could be found thinking of the case of loss of sensors, i.e. one would desire solutions of level *h* such that, if one or more sensors are malfunctioning, the loss of information in the system is as small as possible.

Motivated by these arguments, in (Viti et al., 2014) we introduced a new metric that considers the ensemble of information from both observable and non-observable link flows, with an attempt to measure the amount of information that sensors still provide to the unobserved link flows. To do so, we related the amount of information with the Null Space of the link-route incidence matrix A, which mathematically describes the degrees of freedom resulting from solution under-determinedness. In addition, a metric was introduced to allow ranking of full observability solutions based on a limited number of "families". These families depend on the total amount of information

contained by the different full observability solutions, in terms of quantity and quality of independence-dependence relationships. This metric was then employed in a greedy algorithm to identify an efficient sequence of link flows to measure in order to maximise the information on the network, by identifying and ranking the maximum information loss on the unobserved link flows in relation to a full observability solution containing the same subset of flows.

In route-based partial observability problems the selection of the links to monitor may take advantage of a certain degree of information redundancy at the route level, in contrast with the problem of finding the tight upper bound of the minimum number of sensors, which clearly seeks for routes with theoretically no overlapping links. This will be explained with a simple illustrative example in the next section.

With an attempt to create a bridge between finding linearly independent routes and realistic routes, i.e. where realistic route flows are likely to be observed, in (Rinaldi et al., 2015) the authors proposed a heuristic approach using *k*-shortest paths, a rather conventional method to generate routes in real-sized network, which has the advantage to select routes that are likely containing a good deal of the OD flows, especially in low-congested networks. A variant of the well-known Dijkstra (Dijkstra, 1959) algorithm was used to sequentially generate linearly independent routes for each OD pair. In the proposed greedy *k-independent shortest paths* algorithm (KISP), when a new candidate path is generated, the algorithm verifies its linear independence with respect to the other paths already included in the generated set of paths. Tests on small and mid-sized toy network showed that full observability solutions found using route sets generated with KISP were systematically better than randomly generated route sets, and that more efficient pivoting solutions may be found. This suggests that a set of sensors to be installed for more efficient full observability solutions can still be sought and strongly depends on the way route-link relations are defined.

3. Methodology pt I: exact approaches

As we discussed in Section 1, in this work our objective is that of assessing the relationship between quantity and quality of information embedded in full observability solutions and the composition of the underlying route set. Our hypothesis is that embedding specific independence constraints in the route set generation procedure can be very beneficial in terms of information content, specifically looking at partial observability applications.

In this Section we first provide a classification of information, which will be essential to understand the later definition of independent route sets. Then, we describe our methodology for finding exact independent route sets and we introduce two algorithms for approximate route set generation.

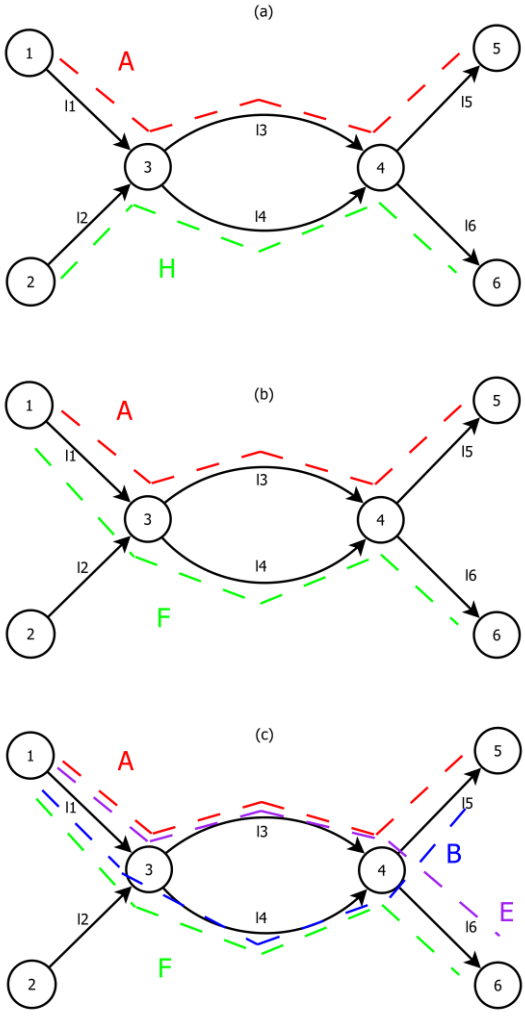
Defining information (in-)dependence

As discussed earlier, link to route relationships form the basic information block in (a specific class of) link inference problems. This information is usually represented through incidence matrices in the form $\mathbf{A} \in \mathbb{R}^{|I|\times|R|}$, where the matrix's columns represent the $|R|$ routes composing the route set R.

In order to understand how different route sets influence observability problems, in this approach we characterize the information brought in by each new route as pertaining to one of the following three sets:

- Non Redundant (NR) set: a route $r_i \in R$ pertains to the NR set if the links traversed by it were not previously included in the route set (i.e. the route consists *solely* of new, previously unavailable *direct measurement* information)
- Redundant while Informative (RI) set: a route $r_i \in R$ pertains to the RI set if its inclusion in the route set allows deriving algebraic interactions between previously non-interrelated links (i.e. the route includes *at least one* new, previously unavailable *direct measurement* information, but isn't solely composed of such information).
- Purely Redundant (PR) set: a route $r_i \in R$ pertains to the PR set if its inclusion in the route set R brings no further information to the system (i.e. the route is entirely composed of links which have already been traversed by any other (combination of) route(s) in the full set).

In Figure 1 we showcase three simple examples of route sets for a small network: Figure 1(a) shows two routes both pertaining to the NR set, in 1(b) we show how a different choice for one of these two routes yields instead RI information, while in 1(c) we show how an additional third and fourth routes would yield PR information.



$A, H \in NR$

$A, F \in [NR \cup RI]$

$A, F \in [NR \cup RI]$

$B, E \in PR$

Figure 1: example of route set with (a) Non-Redundant information only, (b) Non Redundant and Redundant while Informative, and (c) Non Redundant, Redundant while Informative and Purely Redundant.

From an algebraic perspective, this characterization can be classified according to the effects that routes pertaining to other sets have on the rank of the link-to-route incidence matrix A associated with the full route set R . Specifically, the following properties arise from the definitions above:

$$\begin{aligned} R' = R \cup r_i : r_i \in NR \rightarrow rk(A') > rk(A) \\ R' = R \cup r_i : r_i \in RI \rightarrow rk(A') > rk(A) \\ R' = R \cup r_i : rk(A') = rk(A) \rightarrow r_i \in PR \end{aligned} \tag{3}$$

that is, membership of either NR or RI sets is a sufficient condition for guaranteeing that the route is linearly independent from all others; a weaker condition exists on the PR set, stating that all linearly dependent routes must be members of the PR set, but that membership itself is not a sufficient condition.

Example 1(c) clearly shows how membership of the PR set does not guarantee linear dependence: adding either routes B or E to the set $\{A, F\}$ would indeed yield an increase in the rank of the associated link-route incidence matrix from 2 to 3, while adding both yields no further gain.

In general, an arbitrarily generated route set (including the **full** route set) will be a union of all three subsets:

$$R = \{r_1, r_2, ..., r_n\} = [NR \cup RI \cup PR] \tag{4}$$

This specific subdivision in different sets of information allows us to clearly define the route set characteristics through which different goals pertaining to observability problems might be reached.

When seeking to minimize the overall amount of sensors in full observability problems the best possible route set is determined by maximizing Non Redundant information while minimizing both PR (foremost) and RI sets. This is equivalent to determining the *minimum linearly independent route set,* and, as it has been shown in (Castillo et al., 2014), such a solution does indeed yield the smallest possible amount of sensors to be installed, specifically one per each route. The authors proceeded then to extend this set through a heuristic, aiming at including extra information to result in a more realistic representation of overall route choice from a user behavioural perspective.

As we have discussed previously, when dealing with partial observability problems one seeks to maximize how resilient the overall sensing infrastructure can be even when some of the sensors necessary to achieve full observability are missing due to e.g. budget constraints or sensor failure (an in-depth discussion related to how sensor failure / missing sensors influence the error bounds on unobserved links can be found in (Viti et al., 2014)).

In terms of the three information subsets, this translates into optimally balancing the composition of the NR and RI sets, while at the same time minimizing the cardinality of the PR set. Specifically, by maximizing the cardinality of the $[NR \cup RI]$ set the amount of both directly available and indirectly inferable link information is also maximized, as indeed the higher the cardinality of this set and the higher the rank of the corresponding link-to-route incidence matrix.

In this context, balancing the composition of the two NR and RI sets refers to the natural trade-off arising between the respective members: when seeking the maximum cardinality of the joint set, a reduction in the member count of the NR set must be met with a larger increase of members in the RI set. Figure 2 shows such an example, based on that of Figure 1(a), in which route H has been substituted by two equivalent RI routes (F and G).

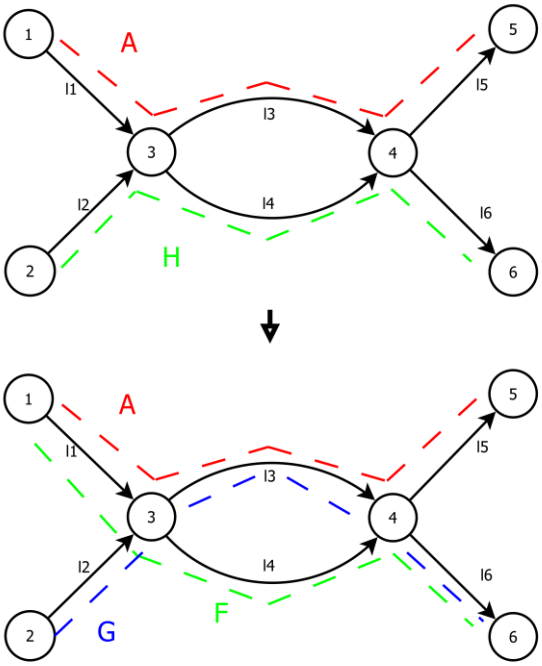


Figure 2: Example of set memebership balancing.

In this work, we develop a methodological framework that constructs route sets in order to meet such optimal (from a resilience perspective) information quality composition. This framework is then adapted to exact and approximate algorithms. As we detail in the next subsection, one of the key advantages of this framework is that the condition of membership to the $[NR \cup RI]$ set can be captured through specific route independence rules, formulated exactly through a quadratic binary programming approach.

Route independence rules

As introduced earlier, determining the minimum linearly independent route set yields a solution where $A = [NR]$ and, ideally, $[RI \cup NR] = \varnothing$ (in general, the two redundant sets should be as small as possible, but might not be feasibly empty depending on the network's topology).

Our intuition is that when extending this initial route set with well-chosen routes, both NR and RI sets will increase in size, while the PR set will remain as small as possible.

Based upon these considerations we form the following hypothesis: "determining the *maximum independent route set* for a given network yields maximal, optimally balanced NR and RI information while minimizing PR information".

To reach this objective we introduce a methodology capable of constructing a set of routes R such that the three following conditions are verified:

i. The set is maximal in terms of cardinality
ii. All routes are independent from one another
iii. All *combinations* of routes are independent from one another

We begin by formalizing our chosen definition of independence between routes, which is based on that introduced in (Castillo et al., 2014).

Given a generic transportation network N described by a directed graph $N \sim G(L,V)$ with L the set of links and V the set of nodes and $R = \{r_1, r_2, ..., r_n\}$ the set of routes, we define independence between routes as follows:

$$\begin{aligned} & r_i = \{l_1, l_2, ..., l_n\} \\ & r_j = \{l_a, l_b, ..., l_z\} \\ & INDEP(r_i, r_j) \leftrightarrow \exists l_k : l_k \in r_i \wedge l_k \notin r_j \end{aligned} \tag{5}$$

given a route r_i composed by a set of links n and a route r_j composed by a set of links z (where $n, z \in L$ and $|n| \leq |L|$ and $|z| \leq |L|$), the two routes r_i and r_j are independent from one another if and only if there exists a link l_k which is included in route r_i and not included in route r_j.

When considering an additional third route r_k, independence between combinations of routes can be defined as follows:

$$\begin{aligned} & r_i = \{l_1, l_2, ..., l_n\} \\ & r_j = \{l_a, l_b, ..., l_z\} \\ & r_k = \{l_\alpha, l_\beta, ..., l_\omega\} \\ & INDEP([r_i \cup r_j], r_k) \leftrightarrow INDEP(r_i, r_j) \wedge \exists l_\mu : l_\mu \in r_k \wedge (l_\mu \notin r_i \vee l_\mu \notin r_j) \end{aligned} \tag{6}$$

that is, a third route k is independent from another independent combination [i, j] if and only if there exists a link l_μ which is included in route r_k and not included in either route r_i or route r_j. In this context, we dub "combination of routes" any set of routes bearing cardinality greater than one, such as the set $[r_i \cup r_j]$.

Following these definitions, the two following properties hold:

$$\begin{aligned} & INDEP(r_i, r_j) \leftrightarrow INDEP(r_j, r_i) \\ & \neg(INDEP(r_i, r_j) \wedge INDEP(r_j, r_k) \rightarrow INDEP(r_i, r_k)) \end{aligned} \tag{7}$$

that is, the independence relation between routes is symmetric, and it is *not* transitive. This latter property is considerably influential on the complexity of the problem at hand: lack of transitivity implies that guaranteeing independence between all possible combinations of routes requires their full enumeration.

This definition of route independence is, as introduced earlier, not equivalent to the linear independence definition used by (Castillo 2014): a set of routes meeting conditions (5) and (6) will also be linearly independent, while the opposite cannot be guaranteed. As we'll though showcase in what follows, our definition allows to obtain a very compact problem formulation, which can be approached from a graph theoretical perspective, internalizing the nonlinearity of the route independence constraints. As far as the authors know, the same simplification cannot be applied to *linear* independence constraints, meaning that a full-fledged nonlinear program would arise, where each possible combination of routes would require explicitly determining the rank of the associated link-to-route incidence matrix, in order to determine the largest possible linearly independent set.

Problem formulation and hypergraph representation

Based upon this notation, our desired route set generation approach can be formulated through the following constrained optimization problem:

$$\begin{aligned} &\max |R| \\ &s.t.\ \ INDEP(c_i, c_j) \forall c_i, c_j \in S_c \end{aligned} \tag{8}$$

where $S_c = \bigcup_{k=1}^{2^{|R|}} \binom{R}{k}$ is the set of all k-combinations of routes r_i populating the full route set R.

Approaching this problem directly is impractical: while route enumeration can be formulated exactly through very simple mixed integer linear programs, the additional independence constraints cannot be directly formulated in linear form, thus requiring ad-hoc solvers and reducing any guarantees of finding a definitive optimal solution.

To address this issue, we instead reformulate problem (8) in graph theoretical terms: based on the route set R, we construct an opportune hypergraph $HG(L_h, V_h)$, whose vertices $v_h \in V_h$ represent **all** *k*-combinations of routes for the original network $G(L,V)$, and whose arcs $l_h \in L_h : l_h = (v_i, v_j),\ v_i, v_j \in V_h$ capture instead the independence relationships between routes and combinations thereof.

We represent the individual hypergraph vertices v_h through a set of $|L|$ elements $v_{h_i} \in \{0,1\}^{|L|}$ which take a value of 1 if the corresponding link is included in the (set of) route(s) pertaining to the given vertex and 0 otherwise. In the case depicted earlier in Figure 1(a), for example, the resulting hypergraph vertices are as follows:

$$\begin{aligned} v_A &= \{1,0,1,0,1,0\} \\ v_H &= \{0,1,0,1,0,1\} \\ v_{AH} &= \{1,1,1,1,1,1\} \end{aligned}$$

Thanks to this specific representation, to which we refer as "bitwise signature", the condition of independence between different routes composing a given route set can be easily captured through a bitwise algebra operation:

$$INDEP(r_i, r_j) \leftrightarrow (v_{r_i} \vee v_{r_j} \neq v_{r_i}) \vee (v_{r_i} \vee v_{r_j} \neq v_{r_j}) \tag{9}$$

that is, a couple of routes (in general, vertices) r_i, r_j are independent from one another if and only if their *combination* (computed through a bitwise OR operation) is different from at least one of the two original parent elements. Accordingly, an arc between two vertices v_h, v_k will be drawn if and only if condition (9) is met. Once more referring to the example of Figure 1(a), we can simply conclude that the hypergraph for the two routes A, H is in fact complete, as condition (9) holds for all couples $[v_A, v_H], [v_A, v_{AH}], [v_H, v_{AH}]$.

This reformulation bears a very attractive property: an equivalent solution for (7) can be determined by solving a constrained form of the *maximum clique problem* on the graph HG, a well-known problem in operations research and graph theory literature for which several formulations and algorithms are readily available (Alidaee et al., 2007; Macambira and de Souza, 2000). Recently, a similar solution scheme was proposed by (Fu et al., 2016) to locate a combination of active and passive sensors in the network.

Specifically, we introduce parenthood constraints to the maximum clique problem to correctly represent condition (iii), i.e. to ensure that *all combinations* of selected routes are independent from one another. This constraint can be formulated as follows: if a vertex pertaining to a *k*-combination of routes is selected, all vertices pertaining to the $[k-1$ to $1]$-combinations of routes composing the original vertex should also be selected.

Through the following theorem we show that, indeed, solving the constrained max clique problem on the hypergraph HG yields a set of routes meeting all conditions (i-iii).

Theorem 1: Finding a parenthood-constrained maximum clique C_m on the *complete* hypergraph $HG(V_h, L_h)$ is equivalent to determining the maximum set of routes R such that all routes and combinations thereof are independent from one another.

Proof:

We begin by proving the independence properties by contradiction:

Hyp 1: Let a clique $C_m : \exists v_j \in C_m : \exists (r_s, r_t) \in v_j : \neg INDEP(r_s, r_t)$ be a solution to the max clique problem formulated on hypergraph $HG(V_h, L_h)$, that is, let a vertex exist in the maximum clique C_m such that two routes r_s and r_t composing it are violating the independence condition as defined in (9).

It follows from the basic properties of a clique that $C_m : HG[C_m] \equiv K_n, n = \mid C_m \mid$, i.e., the subgraph induced on hypergraph HG by the clique must be *a complete graph* with n vertices.

By construction, the vertex set of the hypergraph V_h must be such that the parenthood constraints are met, that is, no less than two vertices representing the 1-combination of routes *s* and *t* must exist: $\exists \{v_s, v_t\} \in V_h : v_s = r_s, v_t = r_t$.

However, the arc set L_h is such that: $\neg INDEP(r_s, r_t) \rightarrow \nexists l_h \in L_h : l_h = (v_s, v_t)$, that is, no arc can exist between any two vertices (s,t) violating condition (9).

This implies that $G[C_m] \neq K_n$, thus violating hypothesis 1 (C_m cannot be a clique *and* contain vertex v_j). □

Through contradiction we can furthermore show that, indeed, the determined solution entails the *maximum* set of independent routes R:

Hyp 2: Let two route sets $R' = \{r_1, r_2, ..., r_n\}$ and $R'' = \{r_1, r_2, ..., r_n, r_{n+1}\}$ both meeting independence conditions (ii) and (iii): the parenthood-constrained maximum clique solution C_m captures route set R'.

A clique C is maximal if and only if the following condition is met: $C : \nexists v_k : HG[\{C, v_k\}] \equiv K_{n+1}, n = |C|$, that is, if there is no vertex v_k such that the clique C_m can be extended and still be a complete graph.

Since $R" = \{R', r_{n+1}\}$ is meeting all independence conditions it follows that $\exists C : HG[C] \equiv K_{n+1}$. However, following parenthood constraints, $|C| = |C_m| + 2^{n-1} > |C_m|$, contradicting hypothesis 2 (C_m is not maximum). □

To form a proper optimization problem, we base ourselves upon the Mixed-Integer Quadratic Programming (MIQP) formulation introduced by (Theorem 2.2 in Bomze et al., 1999). From the hypergraph HG the vertex-to-vertex adjacency matrix A_g can be extracted, defined as follows:

$$\mathrm{A}_g = \{a_{ij}\} \in \{0,1\}^{|V_h| \times |V_h|} \tag{10}$$

where the elements a_{ij} are equal to 1 if an arc l_h connects hypergraph vertices i and j and 0 otherwise. The max clique problem's quadratic matrix Q_{HG} can be derived as follows:

$$Q_{HG} = (\overline{A_g} - I) \tag{11}$$

where $I \in \mathbb{R}^{|V_h| \times |V_h|}$ is the identity matrix and $\overline{A_g}$ the adjacency matrix pertaining to the complement graph $\overline{HG}$, which is computed according to the following equation (12).

$$\overline{A_g} = (J_{|V_h|} - I - A_g) \tag{12}$$

where $J_{|V_h|} \in \{1\}^{|V_h| \times V_h|}$ is the all-ones matrix of proper dimensions.

The resulting MIQP problem is formulated as follows:

$$\begin{aligned} &\min_x x^T Q_{HG} x \\ &s.t. \, x \in \{0,1\} \end{aligned} \tag{13}$$

and its solution vector $x^* = [x_1^*, ..., x_{|V_h|}^*]$ has elements $x_i^* = 1$ in correspondence to those vertices i being part of the maximum clique.

This quadratic problem finds exact solutions to the unconstrained Max Clique problem by exploiting a specific duality property between the Maximum Clique problem on indirect graphs and the Independent Set problem. Given a generic graph G, characterized by a node-to-node adjacency matrix A, rather than solving directly the Maximum Clique problem on the couple $[G, A]$, one can solve the Independent Set problem on the complement graph $\bar{G}$ characterized by the adjacency matrix $\bar{A}$ (as defined in (12)). The Independent Set problem can be formulated quadratically as in (13). Thanks to the duality between G and $\bar{G}$, solving the Independent Set problem on the latter yields a max clique solution in the former.

To represent the parenthood constraints, we extend this formulation by including an additional set of linear inequality constraints, together with a linear component prioritizing the choice of cliques bearing vertices composed by a larger amount of routes:

$$\begin{aligned} &\min_x x^T Q_{HG} x + f \cdot x \\ &s.t. \begin{cases} P_{HG} \cdot x \leq 0 \\ x \in \{0,1\} \end{cases} \end{aligned} \tag{14}$$

where the rows of matrix P_{HG} are structured as follows:

$$\begin{aligned} &P_{HG}[i,\cdot] = \{p_{i1},..., p_{i|V_h|}\} \\ &p_{ik} = -1 : v_k \subset v_i \\ &p_{ii} = |v_i| \end{aligned} \tag{15}$$

This constraint ensures that all k-1 combinations of elements composing vertex v_i are to be selected if vertex v_i is selected, while not posing any restriction on the selection of single elements, provided vertex v_i is not selected. The weighing vector f is computed as follows:

$$f : f_i = -\frac{|v_i|}{\max(|v_i| \in V_h) + 1} \tag{16}$$

that is, the higher the amount of routes composing the vertex v_i, the closer the corresponding value of f_i is to 1, although *strictly* smaller than one, to ensure dominance of the quadratic component.

Equation (17) presents the full formulation for the example of Figure 1(a), in order to better clarify our contribution:

$$\begin{aligned} &x = [x_A, x_H, x_{AH}] \\ &A_g = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \\ &Q_{HG} = \begin{pmatrix} -1 & -2 & -2 \\ -2 & -1 & -2 \\ -2 & -2 & -1 \end{pmatrix} \\ &f = \begin{bmatrix} -1/3 & -1/3 & -2/3 \end{bmatrix} \\ &P_{HG} = [-1 \quad -1 \quad 1] \end{aligned} \tag{17}$$

For which, trivially, the optimal solution is $x^* = [1,1,1]$.

In Figure 3 we show two examples of hypergraphs and resulting max cliques (highlighted in red), both based on the small network featured in Figures 1 and 2. In Figure 3(a) we show the hypergraph related to the three routes {A,B,F}, and how this, clearly, is an instance in which the combination [A,B,F] is dependent. Figure 3(b) shows instead how this hypergraph changes when adding a fourth route {G}, and how, indeed, the solution [A,F,G] meets our independence conditions.

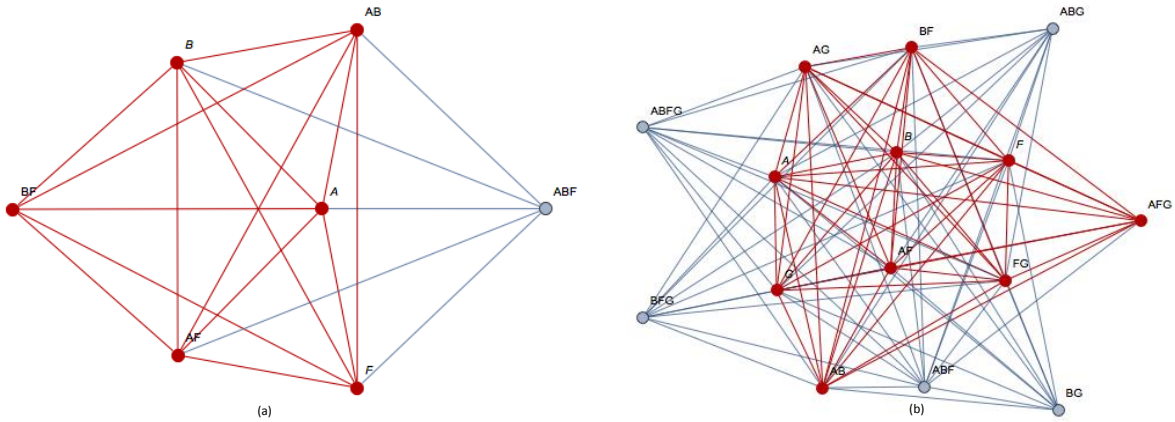


Figure 3: Hypergraph - clique representation of routes [A,B,F] (left) and [A,B,F,G] (right)

Solution algorithms

The hypergraph reformulation is a fundamental transformation of the original problem, which allows us to tackle part of the complexity of determining an exact solution to the maximum independent route set problem: the non-convexities deriving from the independence constraints are explicitly removed from the optimization problem, and instead expressed linearly through the hypergraph's arcs.

The resulting optimization problem (14) can then indeed be easily tackled by commercial optimization software (Such as IBM's ILOG CPLEX or the Gurobi Optimization package), even when dealing with relatively large hypergraph instances.

However, considerable complexity still remains in terms of combinatorial explosion: for a transportation network bearing a total of $|R|$ routes, the cardinality of the set of hypergraph vertices V_h will be $2^{|R|}$, and that of the set of arcs L_h will be, in the worst-case scenario, $\frac{2^{|R|} \cdot (2^{|R|} - 1)}{2}$ (i.e. the hypergraph will be complete).

As the exact problem's complexity increases drastically with the number of routes composing the full route set, we must limit ourselves to seeking exact solutions only for very simple test networks, for which a brute-force approach can be employed to compute the full hypergraph HG.

To address this issue and generalize the methodology towards more significant networks, we hereby develop two algorithms that allow to construct only *portions* of the hypergraph, limiting thus the combinatorial explosion both in terms of computational and, especially, memory requirements. A third algorithm, based on a well-known metaheuristic, is also introduced later in Section 5 to extend these results towards large networks.

As we will show later in Sections 4 and 6, through these algorithms we're still able to deduce insightful conclusions related to our key hypothesis, although sacrificing (in specific instances) the certainty of finding *exact* solutions to problem (8), as the heuristic(s) might fail to generate one or more vertices composing the exact solution.

Three consecutive relaxations characterize the approach presented in the following Algorithms 3.1 and 3.2:

i. Approximation of the initial route set
ii. Exact culling of uninformative vertices during generation (Cul-1)
iii. Approximate culling of bounded-information vertices during generation (Cul-2)

Taking inspiration from our previous works (Rinaldi et al., 2015), we begin by enumerating a base set of k routes R_b for each OD-couple for the given network, therefore directly bounding the amount of nodes characterizing the hypergraph HG by a maximum of $2^{|OD|\cdot k}$. This represents then relaxation i., which we consider "approximated", as the resulting route set is not guaranteed to be complete.

To enumerate this base route set we utilize the well-known *K*-Shortest Path algorithm of (Yen, 1971), which offers two specific benefits to our approach: firstly, due to the inner workings of the algorithm itself, routes successively created for a specific OD will bear very strong similarity with one another, often changing by only a few links. This is beneficial when seeking the *maximum* independent route set, as indeed a strong negative correlation exists between the size of an independent route set and how much variability the single routes exhibit with respect to one another. Moreover, employing the *K*-Shortest Path algorithm might yield better representative route sets, as road users have been shown to consider in their choice sets routes whose differences with respect to the ideal shortest path are very small indeed (for an overview on route choice problems we suggest the interested reader to consult the work of (Prato, 2009)).

Starting from this set we begin by constructing the hypergraph's vertex set V_h, treating the problem sequentially in an OD-by-OD fashion. For each OD, vertices corresponding to each route and each 2-combination of routes are initially added to V_h. After this step, all 2-combinations of the newly introduced vertices and any pre-existing vertex are evaluated and the corresponding vertices are gradually added to the full set V_h. Each new vertex is therefore generated on the basis of either one parent (single route), or two parents (route – vertex combination, or vertex – vertex combination). In the former case the newly generated vertex's bitwise signature is none other than the signature of the route itself, while in the latter case it is generated by performing a single bitwise OR operation between the two parents' signatures. By operating in sequential fashion, and specifically exploiting the associative property of the bitwise OR operation, we avoid direct computation of all k-combinations of routes (totalling $2^{K\cdot|OD|}$), instead limiting computations to an increasing quantity of 2-combinations, which amounts to $|OD|\cdot K\cdot\binom{|OD|\cdot K}{2}$. Once all relevant vertices have been added to V_h, the full arc set L_h can then be directly computed by exploring all 2-combinations of vertices: an arc l_i will be created between two vertices v_h, v_k if and only if the two corresponding bitwise signatures differ by at least one element. This forms relaxation (i), the resulting hypergraph will be *exact* if and only if K is selected large enough to, in fact, enumerate all routes in the network.

We then introduce two culling rules, designed to remove nodes from the hypergraph during generation, when these are deemed unnecessary based upon their information content.

The first rule, dubbed (Cul-1), is triggered during the construction of bitwise signatures for prospective children vertices. The philosophy behind this rule is rather trivial: a tentative vertex whose bitwise signature is entirely equal to either of his parents' is bringing no further information to the system, and will automatically be dependent from one of the two (thus, even if the vertex were to be generated, it would never be part of a clique). Based on this consideration, if a vertex's signature is found to be equal to either of his parents', the vertex is culled.

The second rule, (Cul-2), stems from the same philosophy, but is instead heuristically based on keeping track of best bound values for the amount of routes composing the largest vertex of the hypergraph and the amount of information that vertices provide, computed as the simple sum of their bitwise signatures. Following our key objectives, a solution bearing favourable characteristics in terms of route set independence would entail vertices composed by as many routes as possible, while at the same time bearing as high information content as possible. Therefore, vertices bearing a very high information content (greater or equal to the current best bound value) while at the same time being composed of fewer routes than the current best bound are discarded. This rule, representing relaxation (iii.), is approximate, in the sense that the discarded (culled) vertices cannot be guaranteed not to eventually be part of a clique.

To better clarify the effect of the two culling rules, we construct here a simple example, based on the four routes presented in Figure 1. While the hypergraph for the three routes A, B, F presented in Figure 3(a) would, in fact, remain unchanged if either Cul-1 or Cul-2 – or both, were applied, in Figures 4(a-c) we show instead the impact of applying them where the fourth route H is added sequentially to the existing set ABF, and what the impact is on the corresponding max cliques (d-f).

Indeed, in this instance rule Cul-1 affects neither the size of the hypergraph nor the resulting max clique ((b), (e)), while Cul-2 affects both, by culling the four vertices $v_{AH}, v_{AF}, v_{ABH}, v_{ABFH}$ ((c), (f)). As we will detail later, in general applying Cul-1 alone yields favourable results though at a rather high computational price, while applying both culling rules considerably impacts the computational price while delivering acceptable results.

A more in-depth example assessing how the different culling rules impact this small network is included in Appendix A.

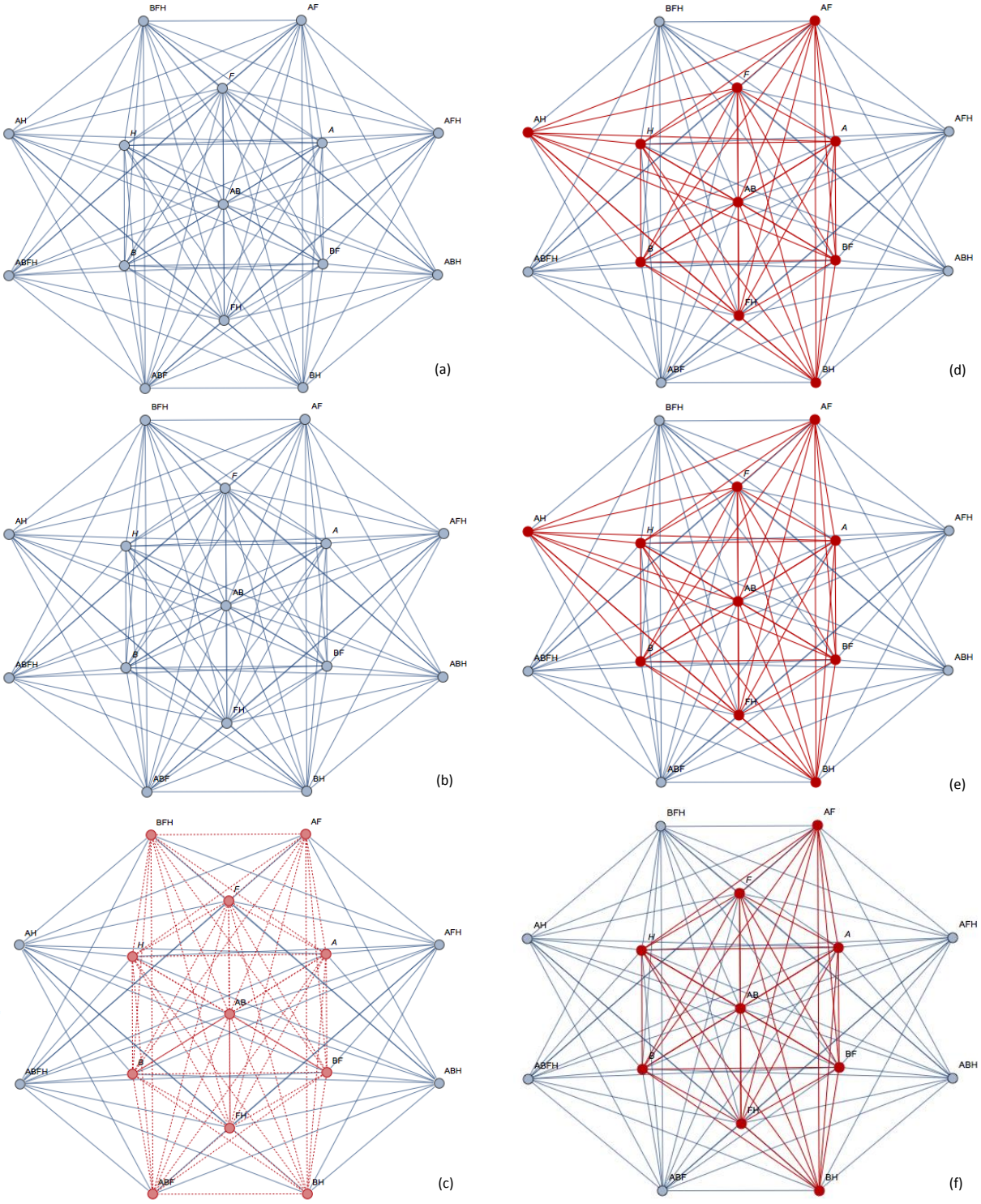


Figure 4: hypergraph representation and effect of applying the two proposed culling rules

We now introduce the two algorithms resulting from the two different culling rules, which will then be employed throughout Section 4.

Algorithm 3.1 (Cul-1 only):

Choose a maximum number of routes per OD couple K

enumerate the base route set R_b
Set $k \leftarrow 1$
for each OD couple odn
  for each route $r \in odn$
    compute bitwise signature for r
    assign to tentative vertex v_k
    for each 2-combination $c = [v_i, v_k]$, $\forall v_i \in V_h$
      compute bitwise signature, assign to tentative vertex v_{k_c}
      (*Cu1-1*) **if** $v_{k_c} \equiv v_i$
        discard v_{k_c}
      else
        add v_{k_c} to vertex set V_h
        Set $k \leftarrow k+1$
  end(**for**)
end(**for**)
for each 2-combination c of vertices $[v_i, v_j] \forall i, j \in V_h, i \neq j$:
  if $v_i \neq v_j$
    create arc $l_h = (v_i, v_j)$;
end(**for**)

Algorithm 3.2 (Cul-1 + Cul-2):

Choose a maximum number of routes per OD couple K

enumerate the base route set R_b
Set $k \leftarrow 1$
Set $bndQt \leftarrow 0, bndIn \leftarrow 0$
for each OD couple odn
  for each route $r \in odn$
    compute bitwise signature for r
    assign to tentative vertex v_k
    for each 2-combination $c = [v_i, v_k]$, $\forall v_i \in V_h$
      compute bitwise signature, assign to tentative vertex v_{k_c}
      (*Cu1-1*) **if** $v_{k_c} \equiv v_i$
        discard v_{k_c}
      else
        Set $currQt \leftarrow | v_{k_c} |$
        Set $currIn \leftarrow sum(bitsign(v_k))$

```
15.          (Cul-2)      if currQt < bndQt & currIn ≥ bndIn
16.                          discard v_{k_c}
17.                       else
18.                          add v_{k_c} to vertex set V_h
19.                          Set k ← k+1
20.                          Set bndQt ← max(currQt, bndQt)
21.                          Set bndIn ← max(currIn, bndIn)
22.       end(for)
23.    end(for)
24. end(for)
25. for each 2-combination c of vertices [v_i, v_j] ∀i, j ∈ V_h, i ≠ j :
26.     if v_i ≠ v_j
27.        create arc l_h = (v_i, v_j);
28. end(for)
```

Depending on the choice for the K parameter as well as on the nature of the underlying network and routes, a hypergraph HG of reduced dimensions is generated by Algorithms 3.1 and 3.2, from which an appropriate quadratic program matrix Q_A can be extracted and fed into problem (11). Under specific conditions, both algorithms might fail to cull any node from the hypergraph HG, yielding therefore the full-sized problem. A computational analysis of such a situation, focusing on assessing the worst-case bound complexity and computational time of the two algorithms, is presented in Appendix D. Unsurprisingly, the probability of the heuristic to build a sufficiently representative portion of the full hypergraph decreases quickly with how small the parameter K is chosen.

Finally, after either algorithm is complete, for the sake of ensuring OD coverage, extra routes might be added even though violating the independence constraint, greedily selected from the originally enumerated route set. As will be shown later, this step does not impact the quality of the final solution.

4. Exact case studies

In this Section we introduce and discuss two case studies based on several small to mid-sized networks, devised in order to validate of our newly proposed methodology and algorithm.

The first case study follows the same structure, methodology and comparisons introduced in our previous works (Rinaldi et al., 2015; Viti et al., 2014), that is, we compare different route generation approaches, including the newly introduced Algorithms 3.1 and 3.2, and *quantitatively* assess their impact in terms of partial observability information content. Results are presented graphically and general performance factors are reported through tables, in order to ensure clarity.

The second case study concentrates on one of these networks, and focuses on obtaining a *qualitative* assessment of which routes have been generated by our algorithm, in comparison with other results available in literature.

The Section is concluded by a discussion of the proposed methodology's limitations, especially in terms of scalability, and introduces the connection between the exact, theoretical methodology of Section 3 and the approximate, heuristic methodology which will be presented in Section 5.

All computational tests have been performed on a 2.4GHz Core i5 processor equipped with 8GB RAM memory. Algorithms 3.1 and 3.2 have been implemented in MATLAB®, while the max clique MIQP problem (11) is solved through IBM ILOG CPLEX optimizer.

Case Study 1: Partial observability information content analysis

In this Subsection, numerical tests are performed on four small-sized networks, with the aim of verifying whether indeed route sets generated following the three conditions stated in Section 3 are beneficial in terms of maximizing the quantity of information embedded in full observability solutions.

Specifically, we employ three different route set generation techniques to construct the link-to-route incidence matrix $A \in \mathbb{R}^{|L|\times|R|}$ needed as an input for link inference full observability problems, and then proceed to solve the full observability problem through Castillo's pivoting procedure, obtaining thus the observability matrices $\Omega \in \mathbb{R}^{|dep|\times|indep|}$. Based on the latter matrices, we then compare the amount of information embedded in each of them, and how this is dependent on the nature of the original route set, in terms of partial observability. The three route set generation techniques we employ are the standard *K*-Shortest Path algorithm (KSP), our previously developed *K*-Independent Shortest Path (KISP) approach and the newly introduced hypergraph based approach of Algorithms 3.1 & 3.2.

We measure the quantity of information embedded in each full observability solution through two metrics, following the same approach of our previous contributions (Rinaldi et al., 2015; Viti et al., 2014): firstly, we compute an a-priori ranking of full observability solutions, based on the observability matrices Ω, as follows:

$$\| \Omega \|_F \cdot rk(\Omega) \tag{18}$$

where $rk(\cdot)$ represents the matrix rank operation. Secondly, we analyse the information from the point of view of partial observability, assessing how quickly information can be collected in the chosen full observability scenario by greedily including sensors to the located set until full observability is reached. Qualitatively, the higher the information content embedded in the full observability solution, the faster/steeper the observation error will decrease when adding links to the observed set.

We now present the four toy networks employed in the case study, shown in Figure 5, and used also in Castillo et al. (2014). The full topological information is instead provided in Table 1.

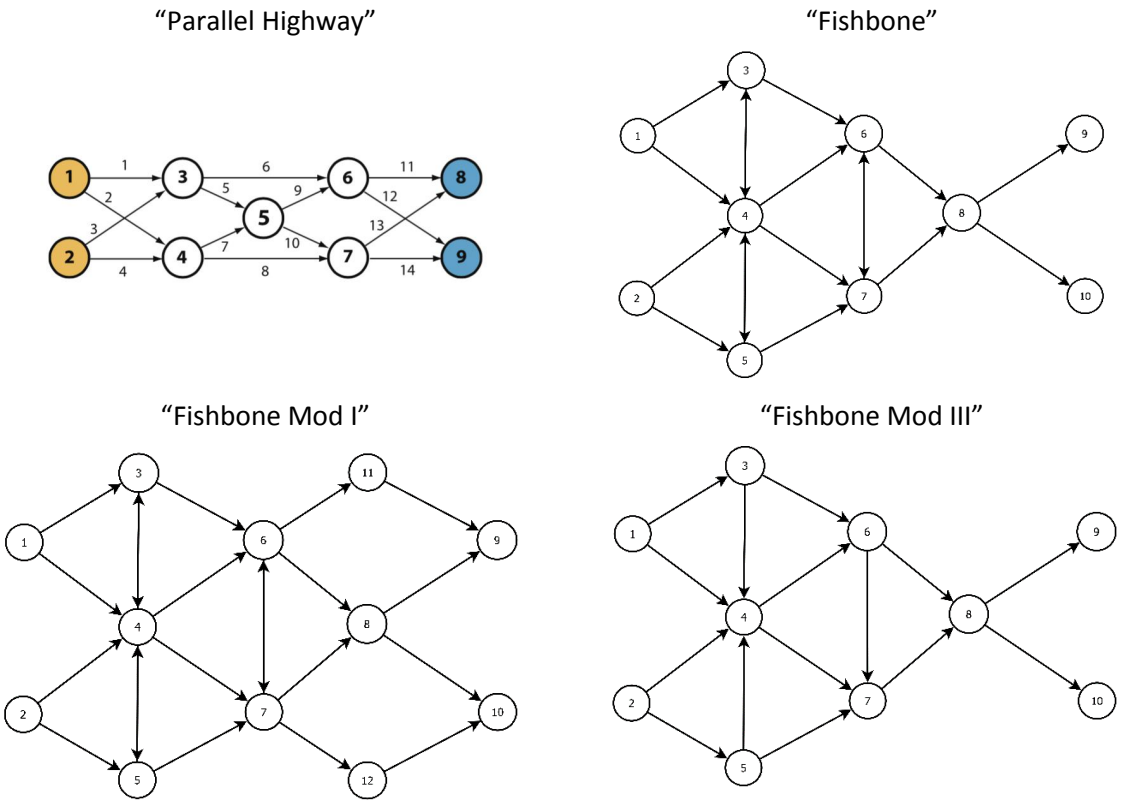


Figure 5: toy networks used to study the impact of route sets in terms of partial observability

As we discussed in our previous work (Rinaldi et al., 2015), when employing either KSP or KISP, link inference problems such as Castillo's pivoting procedure yield non-unique results as different permutations of routes (and thus, route information) can be selected. We capture this stochasticity by enumerating, through the pivoting procedure, 300 full observability solutions for each network. When testing our novel hypergraph-based approach, however, we select just one full observability solution, that corresponding with the first possible permutation, as our objective is indeed obtaining a set of sensors such that the density of information is high enough to avoid the need for further enumeration.

Table 1: topological properties of the toy networks of Figure 7

NETWORK	# NODES	# LINKS	# ODS	# ENUM. ROUTES	# SELECTED ROUTES
PARALLEL HIGHWAY	9	14	4	24	12
FISHBONE	10	18	4	40	12
FISHBONE MOD I	12	22	4	40	12
FISHBONE MOD III	10	15	4	30	12

The comparative results of KSP, KISP and Algorithm 3.1's hypergraph are shown in Figure 6, while Figure 7 presents the tests related to Algorithm 3.2.

Table 2: summary of results for Algorithm 3.1

	Hypergraph Statistics			Solution statistics			
	# Vertices	# Arcs	# Vertices Max Clique	Final routeset size [ind, full coverage]	Tot Memory usage (MB) [RAM + Nodefiles]	Comp. Time (s)	% Gap
Parallel Highway	3583	5991142	405	6, 6	217.77	3600+278.7	4.11
Fishbone	4095	8134666	266	6, 9	275.34	3600+481.59	4.12
Fishbone Mod I	4095	8293427	453	7, 11	270.62	3600+479.92	4.2
Fishbone Mod III	4095	7712078	231	6, 8	288.12	3600+490.11	3.82

Table 3: summary of results for Algorithm 3.2

	Hypergraph Statistics			Solution statistics			
	# Vertices	# Arcs	# Vertices Max Clique	Final routeset size [ind, full coverage]	Tot Memory usage (MB) [RAM + Nodefiles]	Comp. Time (s)	% Gap
Parallel Highway	1918	1819207	388	5, 6	60.17	3600+56.32	2.15
Fishbone	1581	1207671	173	5, 11	42.36	3600+62.09	0.33
Fishbone Mod I	2141	2274412	394	7, 13	75.27	6+154.28	0
Fishbone Mod III	1885	1698224	193	5, 8	60.17	3600+78.65	1.25

For each network, two graphs are shown in Figures 6 and 7. The left hand side figures represent the amount of measurement error, measured through our own NSP metric (18), obtained by successively locating sensors on the network through a greedy heuristic. On each of these, three different data sources are shown: a red distribution of descent patterns, related to the KSP-generated full observability solutions, a green distribution related to the KISP-generated solutions and, finally, a single black dashed line for the newly introduced hypergraph-based algorithms 3.1 & 3.2. The denser the amount of information embedded in a given full observability solution, the faster the amount of error will decrease as new sensors are located, and the fewer the sensors needed to achieve full observability (i.e. a measurement error of 0). On the right hand side figures, instead, similarly coloured histograms showcase the values that the a-priori metric (18) takes for the three different route set generation techniques. Statistics on the generated hypergraph as well as the solution's statistics for Algorithms 3.1 and 3.2 are shown, respectively, in Tables 2 and 3. A maximum computational time limit of 3600s (1h) has been set for the maximum clique solution approach through CPLEX. The corresponding gap in optimality wrt. the unbounded optimal solution is reported in the last column of Tables 2 and 3. The computational times reported in the second to last column are therefore composed of a fixed part related to CPLEX, bounded to a max value of 3600, and a second part which instead is related to the hypergraph nodes and vertices generation.

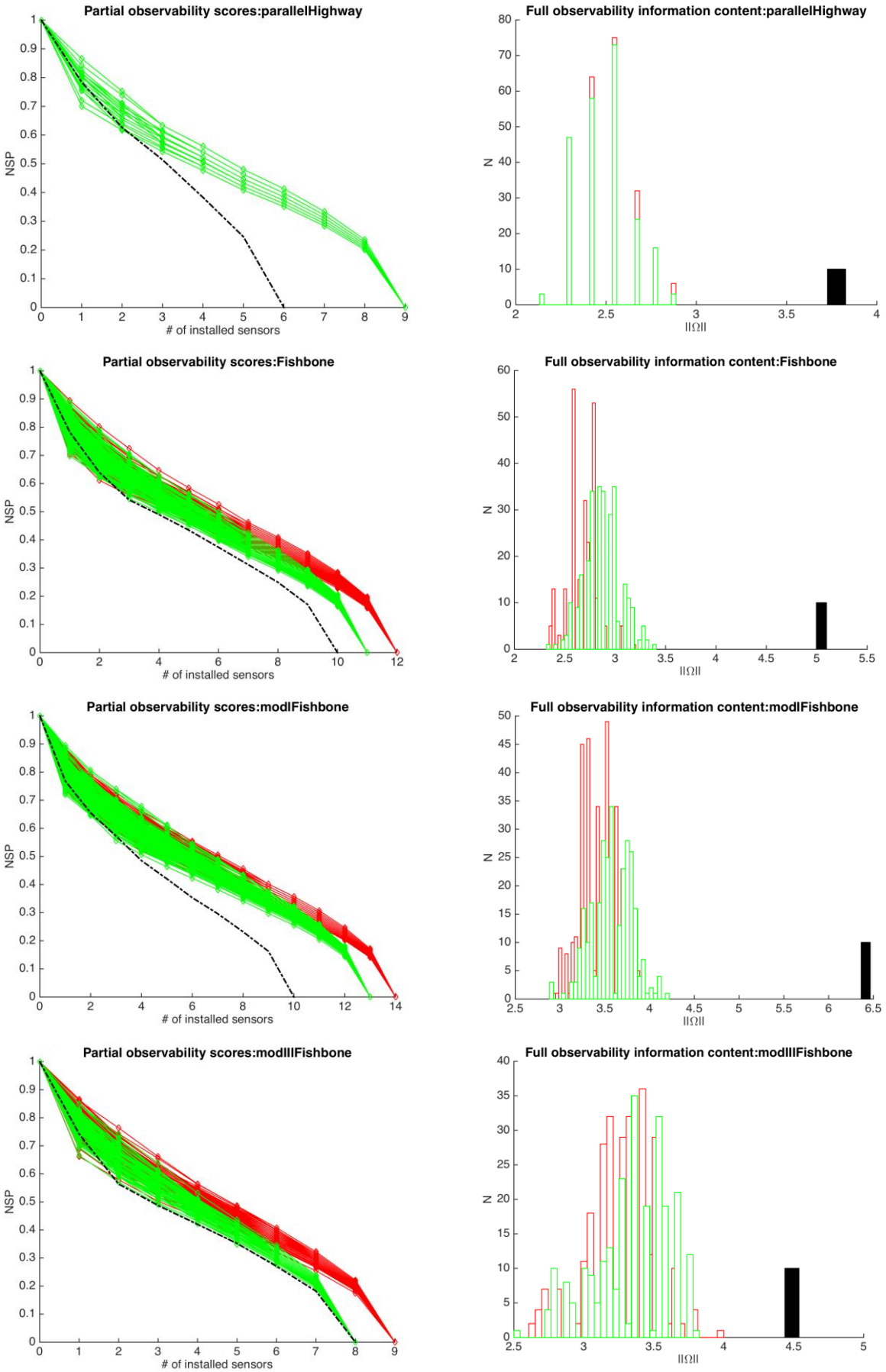


Figure 6: NSP metric results using the greedy heuristic of Viti et al. (2014) and Algorithm 3.1

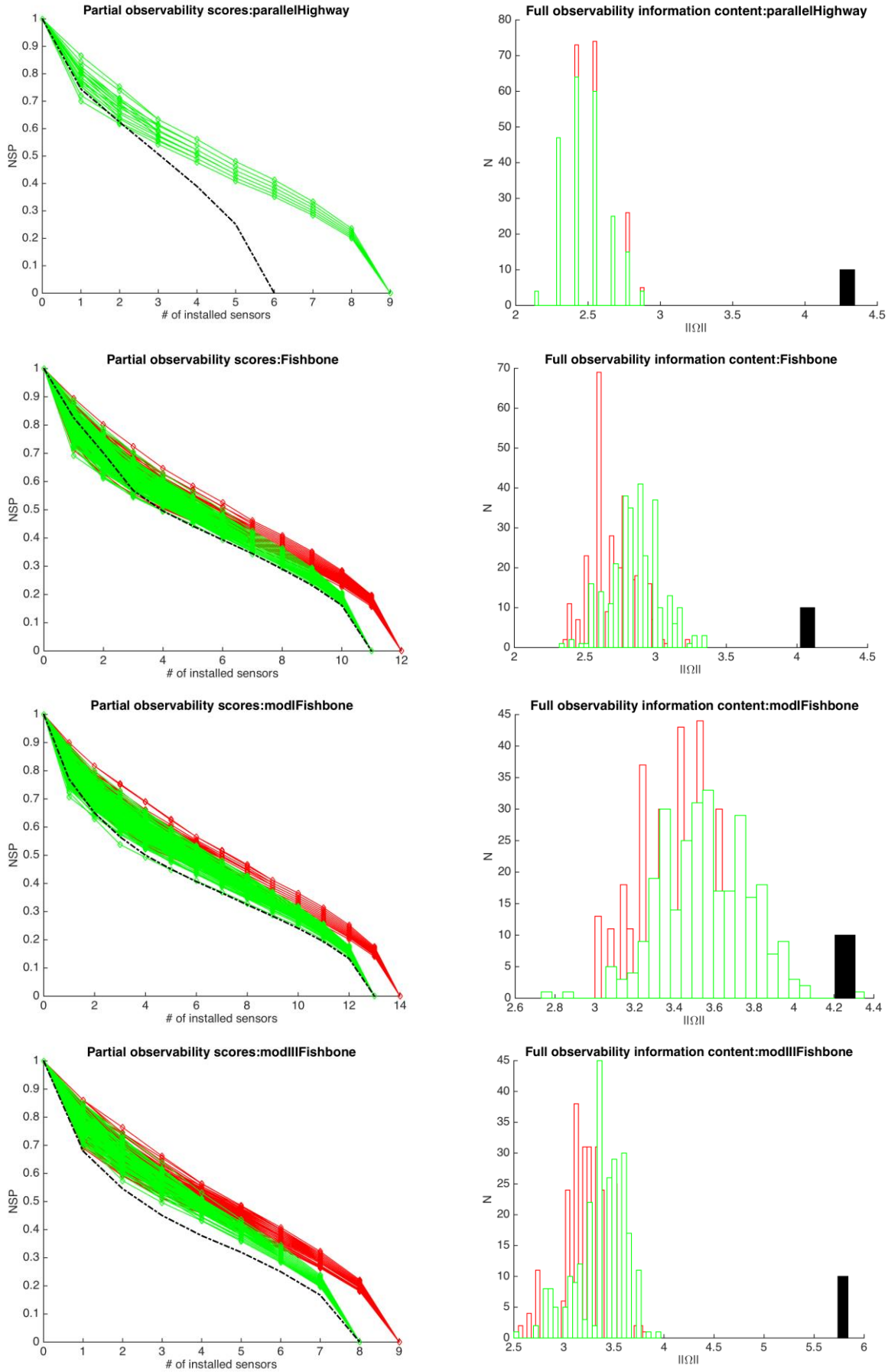


Figure 7: NSP metric results using the greedy heuristic of Viti et al. (2014) and Algorithm 3.2

Analysing the comparative results of our hypergraph-based algorithms compared to the enumerative route set generation heuristics, three main observations can be drawn:

1. The total amount of independent link measures needed to achieve full observability decreases with respect to both KSP and KISP strategies (a condition especially evident for Algorithm 1)
2. The amount of information embedded in the hypergraph-generated full observability solutions is consistently higher than that of the two enumerative strategies, which results in both higher values for the a-priori metric (18) and in very fast measurement error decrease when seeking partial observability solutions.
3. The density of information reached by the newly proposed approach is such that the full observability solution can be computed through a one-shot procedure at little impact in terms of information content, reducing (if not, removing) the need for exploring multiple permutations in the pivoting procedure.

These observations are indeed in line with our initial hypothesis: determining the maximum independent route set yields very high information content from the perspective of partial observability, indeed maximizing the $[NR \cup RI]$ set.

Interestingly, in all the presented instances, keeping the purely redundant information set PR empty appears to be in direct contraposition with ensuring OD coverage. This consideration, in addition to the fact that for all instances only a portion of the full route set was chosen as input for the hypergraph-based component, explains why the partial observability solution identified by the hypergraph-based algorithms is not always dominating in terms of error minimization.

When comparing Algorithm 3.2's solutions with those of Algorithm 3.1, a rather expectable loss of optimality is encountered in most instances, with the notable exception of Mod III Fishbone, where instead, Algorithm 3.2 manages to enumerate a highly informative portion of the network's hypergraph, yielding better overall results. In general, Algorithm 3.2's small loss of optimality appears to be very worth the strong gains in computational speed, while still yielding solutions bearing a consistently higher quality with respect to the two enumerative counterparts KSP and KISP.

Case Study 2: The Parallel Highway network

To underline the importance and relevance of maximizing the Non Redundant and Redundant yet Informative routes to compose the base route set for link inference problems, in this Subsection we perform a comparison between three different solutions, all generated for the "Parallel Highway" network, shown in Figure 8.

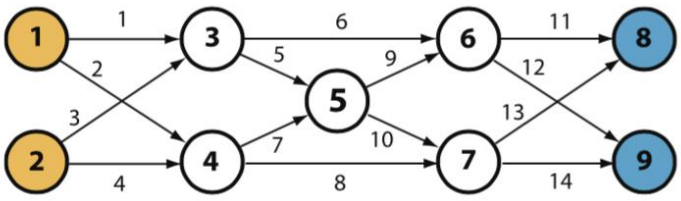


Figure 8: the "Parallel Highway" network

Specifically, we compare three different route generation results: the minimum independent route set, Castillo's link observability/independence heuristic and our maximum independent route set. In this case study, our aim is to characterize the different solutions and their pros/cons in terms of full observability, through *qualitative* analysis.

Figure 9 shows the three different route sets obtained, superimposed on the original network. Different colors represent different routes generated through the given mechanism. In total, 4 routes have been generated by the min independent route set procedure, 9 characterize Castillo's solution

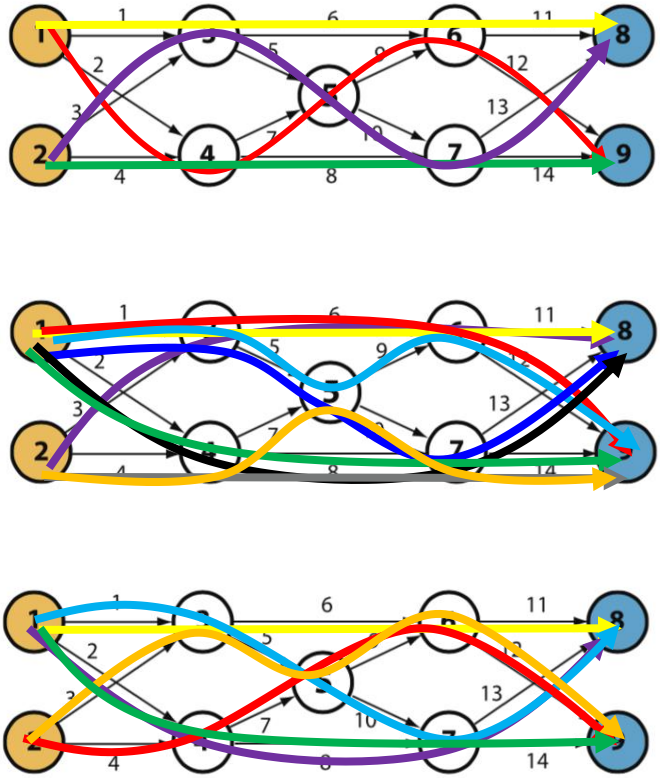


Figure 9: Different route set generation results: Min. Indep (top), Castillo's Indep Heuristic (middle), Max. Indep (bottom)

(as presented in (Castillo et al., 2014)) and 6 compose the max independent route set.

From the point of view of full observability applications of the link inference problem, these three solutions are indeed considerably different.

The solution identified by the minimum independent route set yields indeed the minimum amount of sensors necessary to observe the whole network (4), though at a considerable price in resilience and representability: failure of any of the four sensors equates sudden lack of observations on (up to) 1/4 of the whole network link flows (e.g. a failure on link 2 means no measurements can be inferred on links 7, 9 and 13). Moreover, while indeed minimal, the route set might be under-representative of route choice behaviour, and thus unable to correctly cope with realistic data.

The other two solutions merit a more in-depth comparison, shown in Figure 10:

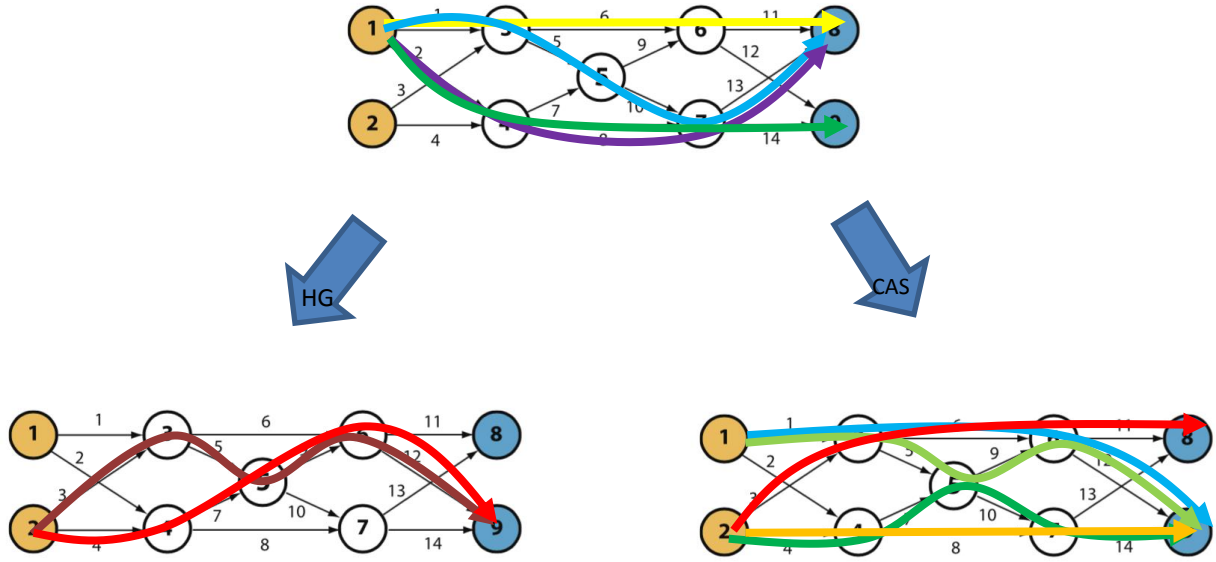


Figure 10: Comparison between route sets generated through Castillo's procedure and our own contribution

As represented in the top portion of Figure 10, both algorithms identify a common route set connecting origin 1 to destinations 8 and 9. This partial set is in itself though insufficient to infer information on all links, and is indeed neglecting links 3, 4, 7, 9, and 12.

As symbolised by the two diverging arrows, from this point on the behaviour of the two algorithms deviates considerably: Castillo's heuristic adds one route separately covering each of the 5 missing links, neglecting any pre-existing relationships introduced by the initial 4 routes. The hypergraph based solution, however, is specifically devised to take direct advantage of these relationships, while maintaining overall independence. The result is a minimalistic addition to the original 4 routes, where indirect relationships between existing and newly introduced routes are exploited in order to infer information about the 5 missing links. This is in turn beneficial in terms of sensor budget, as 6 sensors would be sufficient to infer all network link flows for the given route set, while 9 would be necessary following Castillo's approach (the two corresponding observability matrices are presented in Appendix C). The overall sensitivity to breakdowns is also positively impacted, as partial relationship information can still be exploited to some advantage, as we showed in our earlier studies (Rinaldi et al., 2016; Viti et al., 2014).

Observations

A key limitation to our proposed approach is the inherent curse of dimensionality: two combinatorial problems are embedded one in another, namely the initial route set generation and the subsequent hypergraph node generation. Even when dealing with very small networks such as those presented in this section, unavoidable simplifications had to be taken in order to retain problem tractability.

Employing a subset of *K* paths connecting each OD pair is in itself a very common choice, with limited behavioural repercussions. Indeed, a few studies in literature (Fiorenzo-Catalano et al., 2004; Frejinger et al., 2009) have shown that users are usually limited in their choice to 2-3 "main" route alternatives, for which topological distance (or, more generally, travel time) is the most influential aspect. This simplification is however still insufficient when dealing with large, real-life networks, where the dominating combinatorial aspect will simply shift from the amount of routes to the sheer amount of OD pairs. In these instances, even the more heuristic vertex generation technique of Algorithm 2 will simply prove computationally unfeasible due to excessive memory requirements.

However, we believe that identifying the nature of exact solutions can be source of considerable insight when developing further heuristic applications. In the next Section we present such an approach, in which we build a scalable approximate solution algorithm for the problem (7), by exploiting specific sufficient conditions attributable to exact solutions. As will then be shown in Section 6, this approach can scale up to real-life sized instances by eliminating the need for large combinatorial computations, and moreover, thanks to the introduced sufficient conditions, still yield significant solutions compared to other heuristic approaches.

5. Methodology pt II: heuristic approach

In Section 3 we introduced a formulation for the problem of determining the maximum independent route set (8), together with two possible solution algorithms. As observed throughout Section 4, these approaches are very cumbersome in terms of computational complexity, due to the combinatorial nature of the hypergraph generation procedure. Aiming to scale the promising results obtained for small networks towards real-life instances, in this Section we exploit a well-known metaheuristic, the genetic algorithm, to considerably reduce the overall computational requirements of the approach, while still maintaining a high degree of solution quality.

A Genetic Algorithm approach to maximum independent route set generation

GAs represent a well-known class of optimization metaheuristics, whose inner workings are based upon the process of natural selection (Davis, 1991).

Typically, to formulate an optimization problem in terms of genetic algorithms, the following key elements must be chosen and adapted accordingly to the structure and nature of the problem at hand:

- An atomic representation of candidate solutions for the problem at hand must be adopted, this representation should be as compact as possible and exhibit specific regularity properties, such as fixed length;
- A fitness function, capable of ranking different candidate solutions according to the overall optimality criterion, must be employed;
- A crossover operator (or function) must be selected, which, given two candidate solutions (parents) as input, returns in output a properly generated *genetic combination* of the two (child).

Given these three basic building blocks, the standard GA algorithm “evolves” an initial set of candidate solutions (initial population) towards optimality (higher fitness), by iteratively selecting the most fit candidates from the current population, breeding them through the crossover function and introducing (a selection of) the resulting children into the general population.

Specific criteria (such as which percentage of population is maintained from one generation to the following, or elitism) as well as genetic operators other than crossover (such as the mutation operator) can also be included in the meta-heuristic configuration, depending on how fitting the different facets of natural selection are to the problem at hand. To maintain a good amount of control over the evolution of population and its computational impact, several parameters can moreover be tuned to represent the necessary safeguards: first and foremost, setting a maximum total population constraint directly limits how large the problem of selecting and breeding parents to obtain new children is. Optimal values should be selected such that the population is large enough to thrive over successive iterations (i.e., the portion of the total solution space being explored should be large enough to contain significant solutions), while at the same time small enough to maintain computational feasibility. Another key parameter is the so-called stagnation threshold: as the

number of generations progresses, it is likely to encounter situations in which, from one generation to the next, the maximum overall fitness remains constant. If this condition is consistently met over several successive generations, this indicates that the algorithm has evolved towards a (local) minimum from which it can no longer escape. This threshold represents then a termination condition, stating that after a given number of stagnant iterations the algorithm can be considered complete. Finally, a maximum limit to the total number of generations can also be set, limiting the total time the algorithm will spend looking for minima.

Compared to standard optimization schemes, this heuristic has been found considerably successful when dealing with problems of a discrete nature, such as integer or binary programming problems, as its nature allows to explore a limited portion of the problem's solution space, yielding considerable savings in terms of computational expenses. While the final solution's optimality cannot be guaranteed, as with any heuristic approach dealing with non-convex solution spaces, this methodology has been applied consistently with competitive results in several fields, including some notable transportation applications, e.g. traffic signal timing optimisation (Teklu et al., 2007) and coordination (Putha et al., 2012), bus transit route network design (S. B. Pattnaik et al., 1998), railway systems (Nachtigall and Voget, 1996), etc.

As discussed in Section 3, to obtain exact solutions for the maximum independent route sets we generate a hypergraph HG capturing independence conditions through combinatorial enumeration, after which the problem collapses to a specific variant of the max clique problem on generic, non-directed graphs.

Through a GA approach, we seek to unify these two distinct phases (hypergraph generation and max clique solution) by directly generating only those vertices $v_h \in V_h$ belonging to a clique.

We begin by formalizing the three key components of GA optimization characterising our specific problem instance.

Genetic representation

Each vertex $v_h \in V_h$ pertaining to the hypergraph HG is, in our original formulation, described through a bitwise signature. This representation can be directly employed as the genetic representation for the GA candidate solutions: it is indeed atomic, since it enables a direct mapping of the entire solution space of possible route combinations, and it is of fixed length, since the number of elements in any vertex $v_h \in V_h$ is exactly $|R|$ for a given network.

Fitness function

Given the objective of determining a *maximal* route set, a natural choice for the fitness function is as follows:

$$g(v_h) = \sum_{i=1}^{|R|} v_{h_i} \tag{19}$$

which is indeed maximized by those vertices v_h bearing the highest amount of routes.

Crossover operator

Thanks to the bitwise representation of routes and combinations of routes, a very simple definition for the crossover operator can also be straightforwardly obtained:

$$c(v_M, v_F) = v_M \vee v_F \tag{20}$$

that is, given two parent vertices v_M and v_F , the generated child vertex is obtained through a single bitwise OR operation.

These three basic ingredients, accompanied with proper choices for the main GA parameters, are indeed sufficient to generate a GA capable of enumerating vertices pertaining to the hypergraph HG. However, an essential element is still amiss: to generate solutions for the max clique problem also the independence constraints, captured by the arc set L_h , must also be introduced.

This issue could be approached directly through population control, by ensuring that, at each GA iteration, only children who don't violate said independence conditions are included in the population. However, such a choice would only mildly reduce the combinatorial nature of the original problem. Instead, as detailed in the next paragraph, we introduce specific generation requirements at the level of the crossover operator, such that vertices violating a necessary condition related to independence will not be included in the hypergraph. While approximate, this condition indeed allows to considerably decrease the overall computational burden, as it is applied in a pairwise fashion during crossover, rather than combinatorically over the entire population. To complete the algorithm, a filtering technique can then applied to the final population, to ensure that the output is indeed a clique in the original hypergraph HG.

Necessary condition for route set independence in max-clique solutions

Exact solutions to problem (8) must, by definition, meet the following necessary and sufficient condition:

$$INDEP(v_i, v_j) \forall i, j \in C \subset V_h, i \neq j \tag{21}$$

that is, all couples of vertices found in a clique C are independent from one another.

In our GA based heuristic, we employ a weaker condition, which does not require extensive combinatorial exploration of the full vertex set V_h and is instead formulated and applied for any triplet of child and parent vertices v_C, v_M, v_F:

$$v_C = v_M \vee v_F : \neg(v_M \rightarrow v_F), |\, v_F \,| \geq |\, v_M \,| \tag{22}$$

that is, a child vertex v_C will only be generated as combination of two parent vertices if:

- the two parent vertices are chosen such that the father vertex is always as large or larger than the mother vertex, and
- the mother vertex is *not* a subset of the father vertex (that is, inclusion of the mother vertex introduces new information).

Triplets of vertices meeting condition (22) will, by definition, exhibit the following property:

$$INDEP(v_C, v_M) \wedge INDEP(v_C, v_F) \wedge INDEP(v_M, v_F) \tag{23}$$

however, condition (22) represents only a necessary condition compared to the stricter independence nature of (21). A simple set-theoretical example showcasing this effect can be found in Appendix B.

Full Algorithm specification

Our proposed GA formulation to approximately solve problem (8) is hereby introduced in its pseudocode form.

Algorithm 5.1 (GA for maximum independent route set generation):

Choose a maximum number of routes per OD couple K, a maximum population threshold N_{pop} , an elitism ratio η , a maximum generation threshold $mGen$, a maximum stagnation parameter σ .

1. **enumerate** the base route set R_b
2. **generate** the initial population $pop \leftarrow V_{R_b}$
3. **Set** $V_M \leftarrow pop$
4. **Set** $n_{gen} \leftarrow 1$, $st_{gen} \leftarrow 0$
5. **while** $n_{gen} \le mGen$ and $st_{gen} \le \sigma$
6. **select** $\eta \cdot |pop|$ father vertices V_F from population $|pop|$
7. **generate** children vertices $V_C = c(V_F, V_M)$
8. **if** $\max(g(pop)) == \max(g(V_C))$
9. **Set** $st_{gen} \leftarrow st_{gen} + 1$
10. **Set** $n_{gen} \leftarrow n_{gen} + 1$
11. **update** population pop
12. **purge** final population pop, removing dependent vertices

The algorithm begins by generating the initial population, composed of all single-route vertices corresponding to the base route set R_b . The set of mother vertices V_M is also equally composed, and its composition stays fixed throughout the rest of the algorithm.

The main loop (steps 5 – 11) is responsible for evolving the initial population towards higher fitness values. After selecting a candidate set of father vertices V_F , suitable children V_C are generated by randomly polling couples of parents $v_M \in V_M, v_F \in V_F$ and applying the crossover function $c(\cdot,\cdot)$. A large amount of tentative children is generated in this phase (10^4), with the crossover function taking care of discarding a priori those combinations violating condition (22). A portion of the surviving tentative children is then included in the general population pop in step 11, this portion dependent on the available slots in the general population:

1. **if** $|pop| + |V_C| \le N_{pop}$
2. **Set** $pop \leftarrow [pop \cup V_C]$
3. **else**
4. **discard** $\eta \cdot N_{pop}$ low-fitness vertices from pop
5. **add** $\eta \cdot N_{pop}$ vertices from V_C

that is, all children $v_C \in V_C$ will be added to the general population pop only if there is enough room left with respect to the maximum population threshold. Otherwise, only a specific amount

(dictated by the elitism threshold η) will be added to the population, at the cost of removing lower fitness vertices.

Finally, in step 12, all 2-combinations of vertices composing the final population pop are explored, and couples violating condition (21) removed. The final solution is then, under specific conditions detailed in the next Subsection, guaranteed to be a clique in the original hypergraph HG, although not necessarily a maximum clique.

Discussion

With the objective of keeping computational costs at bay, several simplifications have been introduced throughout the GA application in comparison with the more rigorous approach of Section 3. Specifically, three key steps and choices affect the overall exactness and quality of the final solution obtained by the GA approach: the maximum population threshold, the elitism parameter and the final population purging.

The impact of the first on the final exactness is rather trivial: any choice lower than $N_{pop} = 2^{|R|}$ implies that only a portion of the complete solution space can, at any time, be explored. The lower this parameter is chosen and the lower the chances that high quality solutions will arise. The elitism parameter directly interacts with this choice: the mechanism of population update (step 11) sacrifices an amount of lower ranking vertices in order to meet the total population constraint. This affects the exactness of the approach in two distinct ways: (i) not all acceptable children vertices V_C are included in new generations, directly influencing the direction in which the total solution space is explored and (ii) some older generation vertices are sacrificed over iterations. The latter phenomenon indirectly introduces a systematic error component in the final solution: step 12 performs a combinatorial purging to remove vertices that violate condition (21), but this step's output will be exact if and only if, for any vertex $v_i \in pop$, all of its parent vertices (as defined in eq. 15) are also included in the final population. A sufficiently aggressive $[N_{pop}, \eta]$ combination will violate this condition, thus yielding inexact solutions, as vertices not pertaining to a clique in the original, full hypergraph will *appear* to be independent, due to lack of information.

The reduction of computational complexity is however remarkable: the maximum amount of vertices generated by the GA is limited to $|N_{pop}|$ (as compared to $2^{|R_b|}$ in the exact approach), their generation procedure bound by $O(mGen \cdot |N_{pop}|)$, rather than $O(2^{|R_b|})$. The computational effort of enumerating the corresponding hypergraph arcs is also reduced from $\binom{2^{|R_b|}}{2}$ combinations to $\binom{N_{pop}}{2}$.

In the next Section, we showcase how this GA approach compares to the exact approach presented in Section 4, both in terms of solution quality and computational times. Successively, we approach two mid-scale real-life networks, and analyse Algorithm 5.1's results wrt. other route set generation approaches in terms of full and partial observability performances.

6. Approximate case studies

In this Section we validate our proposed GA based heuristic through two further case studies. Case Study III is based on the same small networks tested in Section 4, and is aimed at comparing the performances of the exact approaches presented therein with those of the proposed approximate heuristic. Finally, Case Study IV presents results related to two mid-size networks, the well-known Sioux Falls network and the sub-urban network of Rotterdam, the Netherlands, following the same structure as Case Study I.

For all tests, the following configuration was chosen for the GA parameters:

Parameter	N_{pop}	η	σ	$mGen$	K
Value	10000	0.1	10	5000	3

Future research will include investigating the approach's sensitivity to changes in these parameter values, especially wrt. the projected solution quality.

Case Study III

We validate Algorithm 3 on the four small networks presented in Figure 5. Rather than performing a full comparative exploration as we did throughout Case Study I, we focus here on comparing the result statistics for Algorithms 3.1 and 3.2 (presented in Tables 2 and 3) with those obtained by applying Algorithm 5.1 on the same instances. Table 4 reports the statistics of Algorithm 5.1.

Table 4: summary of results for Algorithm 5.1

	Generated Solution Statistics			Solution statistics		
	# Vertices	Tot. GA Generations		Final routeset size	Tot Memory usage (MB) [RAM]	Comp. Time (s)
Parallel Highway	9	23	-	7	0.02	7.6
Fishbone	11	59	-	11	0.03	10.01
Fishbone Mod I	12	59	-	12	0.04	11.44
Fishbone Mod III	9	44	-	9	0.02	7.11

Comparing the three tables, we can conclude that the size of the generated route set is consistently larger than comparable exact solutions, meaning that indeed the GA was not able to determine exact solutions for either case. However, the solution quality is still in line with that of Algorithm 3.2, while computational and memory expenses have been drastically reduced.

To conclude this case study, in Figure 11 we compare side by side the results obtained for the parallel highway network in terms of partial observability descent for Algorithm 3.1 and Algorithm 5.1.

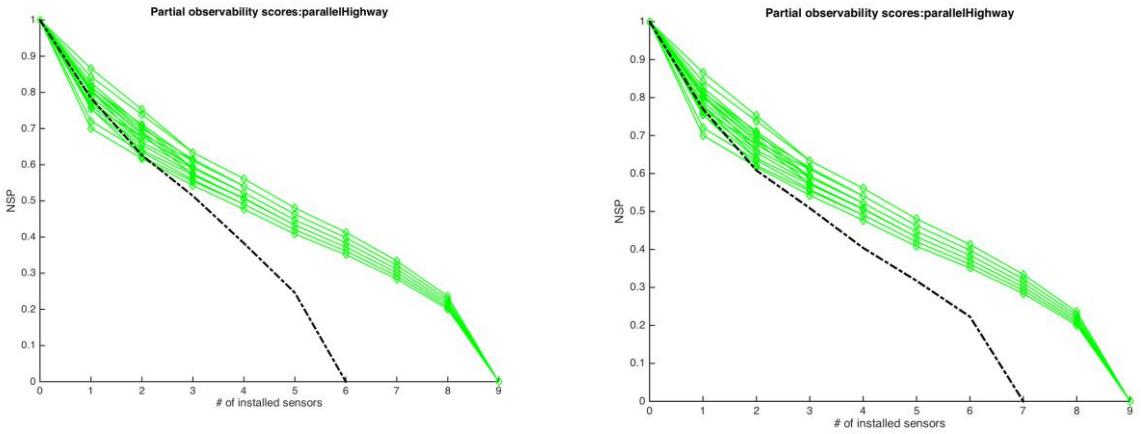


Figure 11: Partial Observability descent results on Parallel Highway. Algorithm 3.1 (left), Algorithm 5.1 (right).

While the solution identified by Algorithm 5.1 is clearly suboptimal wrt. that found by Algorithm 3.1, the result is still very satisfactory when compared to both basic enumeration approaches (KSP, KISP).

Case Study IV

Through this final Case Study, our aim is to validate the theoretical insights developed throughout this work to larger network instances. Specifically, we aim to showcase how generating routes that, at a best effort, adhere to the route independence rules introduced in Section 3 is beneficial to the problem of link flow inference, by reducing the amount of variables needed to attain full observability solutions and, at the same time, yielding information-rich solutions from the point of view of partial observability.

Following the same structure as Case Study I, we quantitatively assess how different route set generation approaches influence full observability matrix Ω for the two selected networks, and how this in turn quantitatively influences partial observability information content. In addition to the already presented KSP and KISP route set generation strategies, we also showcase how Castillo's algorithm for generating linearly independent paths (Algorithm 1, Castillo et al. (2014)) performs from the point of view of partial observability information content. Since Algorithms 3.1 and 3.2 are not applicable to these networks due to dimensionality constraints, no considerations on how close to exactness the result of Algorithm 5.1 is can be drawn at this stage.

The two networks are presented in Figures 12 and 13, while Tables 5 and 6 recap the main network characteristics. The initial route set R_b has been computed, for both, setting $K = 2$.

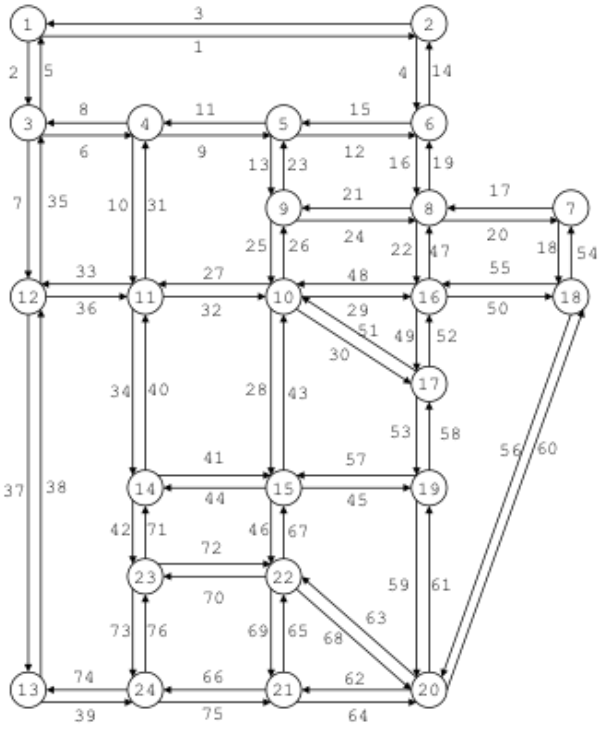


Figure 12: Sioux Falls simplified network

Table 5: Network Characteristics

# NODES	24
# LINKS	76
# OD PAIRS	30
# ROUTES	60

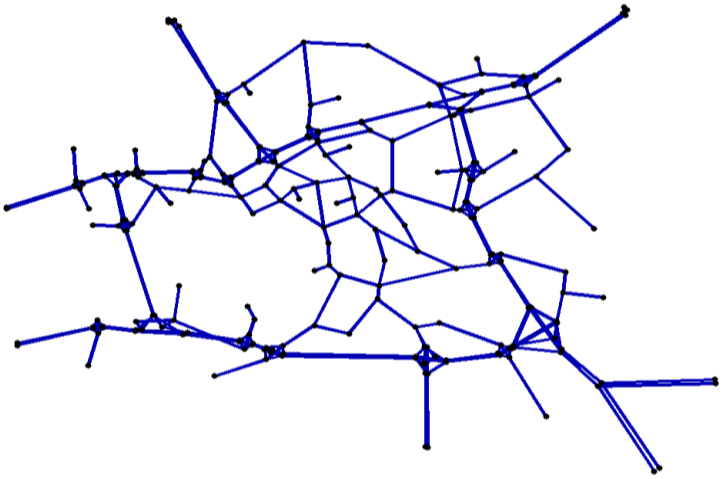


Figure 13: Rotterdam sub-urban network.

Table 6: Network Characteristics

# NODES	243
# LINKS	476
# OD PAIRS	1890
# ROUTES	3779

The comparative results featuring the four route set generation strategies are shown in Figure 14, for the Sioux Falls network, and Figure 15 for the Rotterdam network.

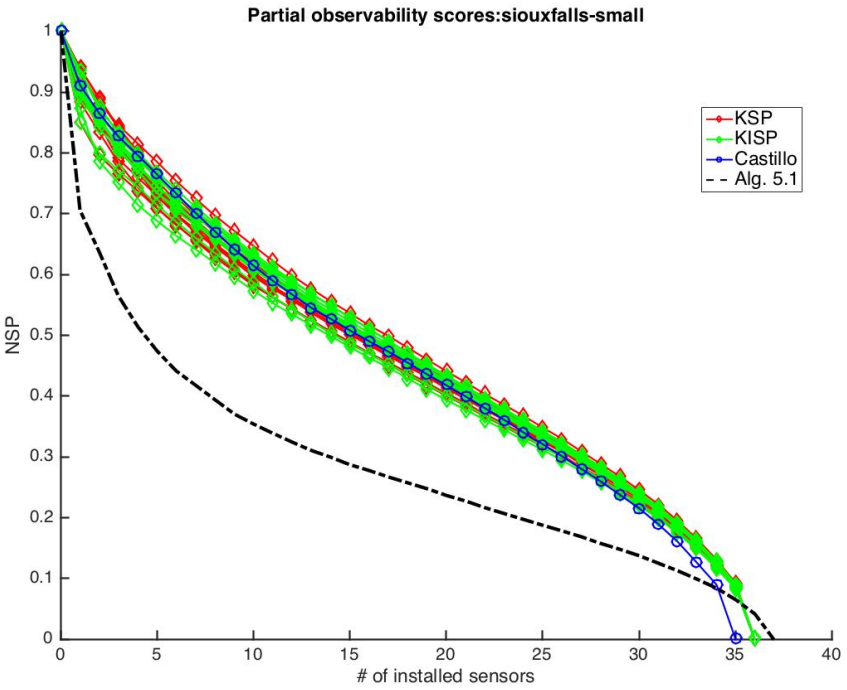


Figure 14: Partial observability information comparison for the Sioux Falls network.

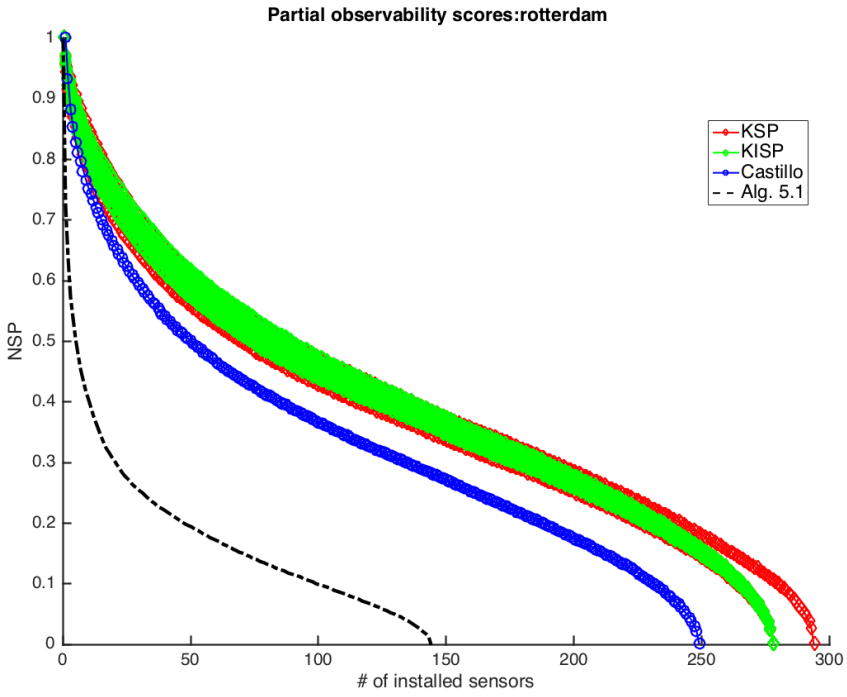


Figure 15: Partial observability information comparison for the Rotterdam network.

In both Figures, the partial observability information gains obtained by incrementally installing sensors based upon minimizing our NSP metric are shown (KSP in red, KISP in green, Castillo's

Algorithm 1 in blue and Algorithm 5.1 in dotted black). Table 7 reports the corresponding computational times for the two networks.

Table 7: Summary of results for Algorithm 5.1

	Generated Solution Statistics			Solution statistics		
	# Vertices	Tot. GA Generations		Final routeset size	Tot Memory usage (MB) [RAM]	Comp. Time (s)
Sioux Falls	37	165	-	37	0.72	281.9
Rotterdam	144	541	-	150	98.26	12259.1

For both networks, the max independent route set solution found by Algorithm 5.1 is clearly dominating in terms of partial observability descent, displaying a very steep initial information gain. Interestingly, while for the Sioux Falls network the final solution in terms of amount of sensors ranges in the same order of magnitude as those obtained by other approaches, for the Rotterdam network this number is significantly lower, amounting to approximately half that needed by the purely enumerative KSP and KISP.

To further investigate this last result, in Figures 16-19 we show a cross-comparison between the four different solutions. We install the first 50 sensors as identified for the four cases (highlighted in red), and show which links in the network become, through the given route-link relationships, partially observable (highlighted in yellow) and fully observable (highlighted in green).

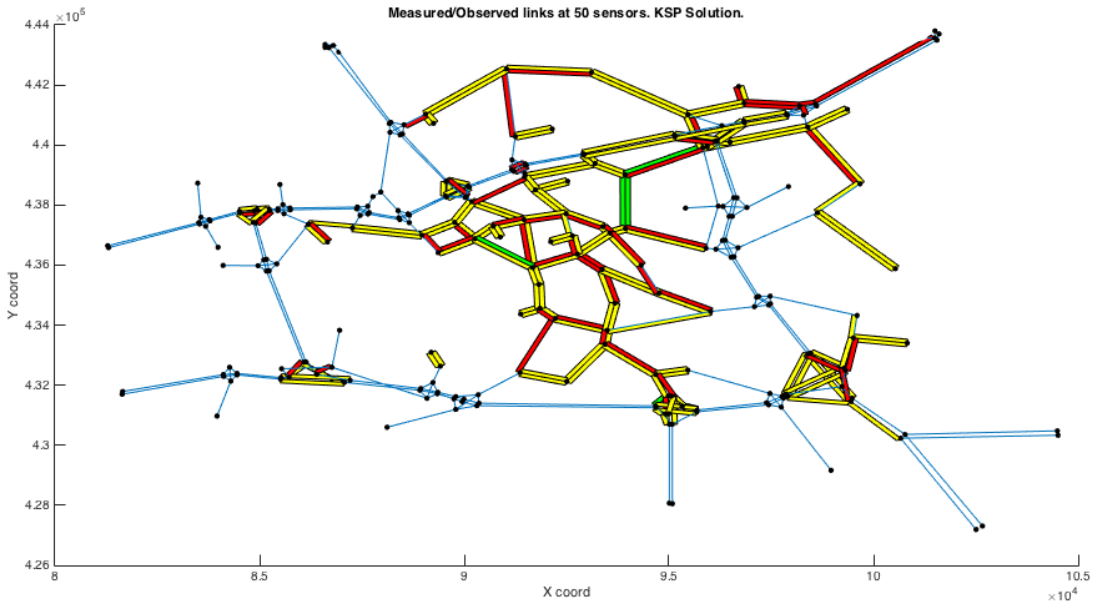


Figure 16: Measured / Observed / Partially observed links for the KSP full obs. solution

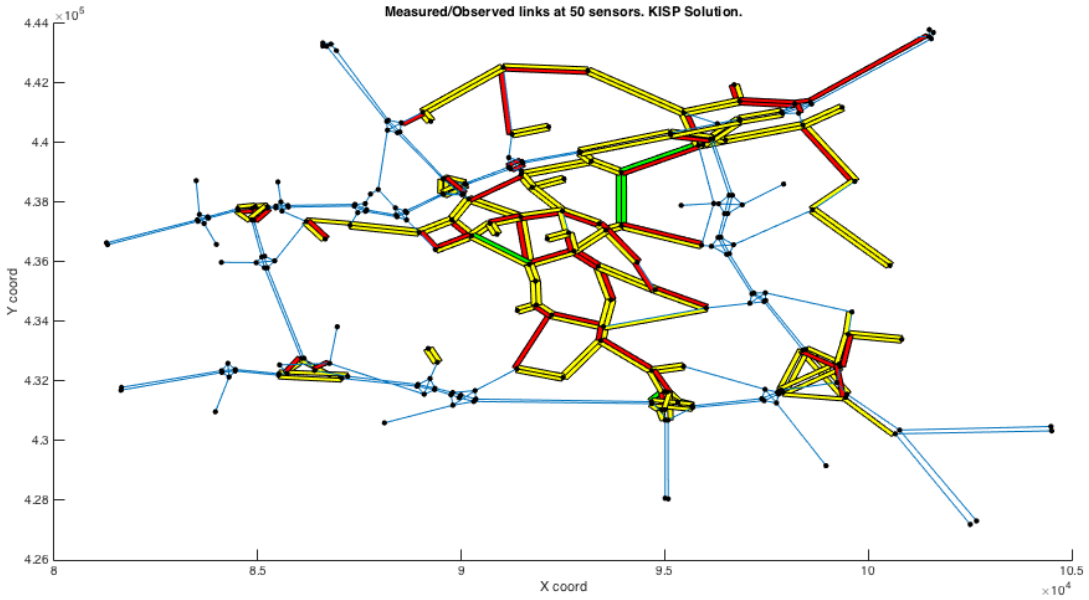


Figure 17: Measured / Observed / Partially observed links for the KISP full obs. solution

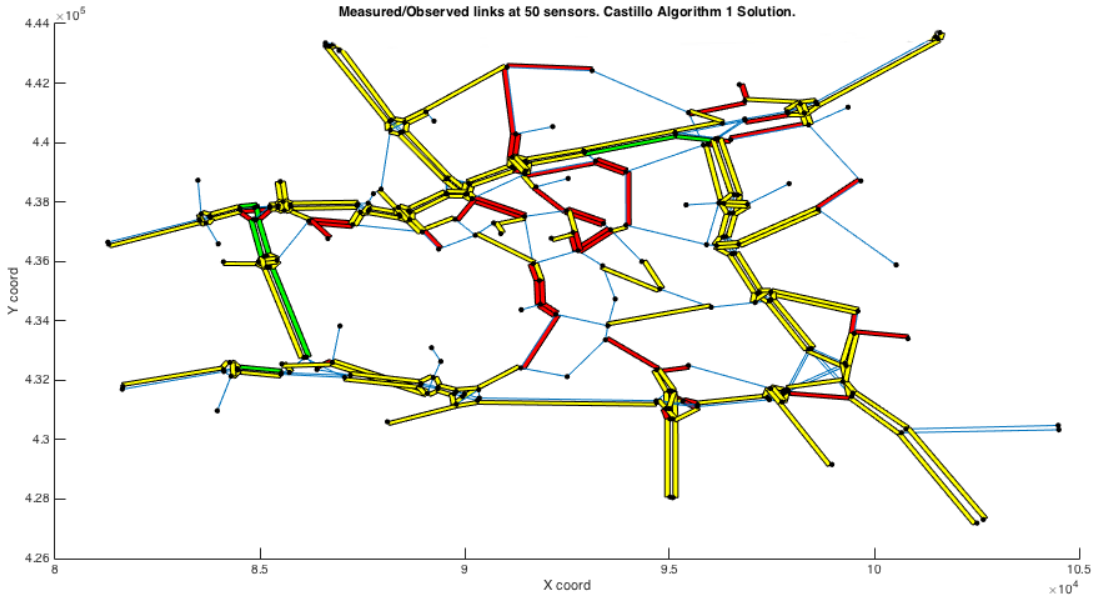


Figure 18: Measured / Observed / Partially observed links for Castillo's Alg. 1 full obs. solution

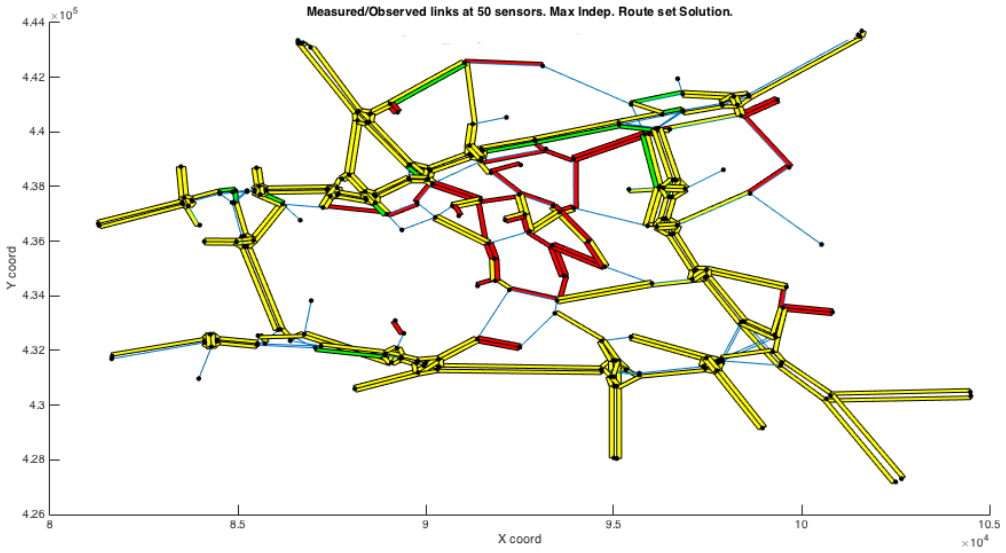


Figure 19: Measured / Observed / Partially observed links for Algorithm 5.1's full obs. solution

As can be seen by comparing the different results, the impact of enumerating route sets bearing denser information is consistently noticeable, as equipping an equal amount of sensors yields considerably more partially and fully observed links. From a geographical perspective, it's interesting to notice how Castillo's and our approach share a considerable amount of links, both focusing on measuring first the city centre close to the OD couples, while the two enumerative approaches have a lower focus, selecting links spanning the whole network.

From the point of view of link flow inference, it's important to stress that while these results are very promising from an algebraic perspective, the impact of discrepancies between the generated route set and the revealed route set certainly requires further investigation.

7. Conclusions

In this paper we studied the impact of route set generation in route-based sensor location problems. We started by providing a definition of information redundancy, by classifying routes as non-redundant, redundant while informative, and purely redundant and by relating these definitions to the concept of independent route sets.

We then formulated a general optimisation problem aiming at maximising the amount of non-redundant and redundant while informative information, and in the same time at minimising the purely redundant one. Exact solutions can be calculated by seeking for the maximum clique of an equivalent hypergraph consisting of all combination of possible routes in a network as nodes, and where the arcs represent the independence relations.

Exact solutions however rely on the construction of the complete hypergraph, which is not computationally feasible even for relatively small-sized networks. For this reason, two solution algorithms have been proposed, which explore specific subsets of the hypergraphs. These algorithms have been shown to efficiently identify solutions for different toy networks. The identified routes sets have the desirable properties of finding through standard pivoting procedures very efficient solutions in terms of full observability, as well as consistently finding information-rich solutions in terms of partial observability.

To validate and extend these results towards large, real-life sized networks, a Genetic Algorithm based solution to the maximum independent route set problem. Thanks to the considerable savings in computational effort, results pertaining to two mid-large scale networks were obtained, further confirming how route set selection criteria play a major role in generating information-dense full and partial observability solutions.

Possible future research directions include exploring the advantages of the proposed approach to applications such as state and OD demand estimation. In previous works we already explored the interaction between optimal sensor locations and data dependent applications such as OD demand estimation techniques. Pursuing this direction, a very interesting topic would then be then evaluating the hypergraph generated route set's realism, and how thus the flows captured by the located sensors influence OD estimation.

Finally, a key research question that still needs proper addressing is assessing how discrepancies between the generated route set and the revealed route set would affect state estimation procedures, with a specific focus on the link flow inference problem.

Appendix A

In this Appendix we validate our proposed independent route set generation algorithms 3.1 and 3.2 through a simple network introduced in our previous works, dubbed “Candy”, shown again in Figure 20.

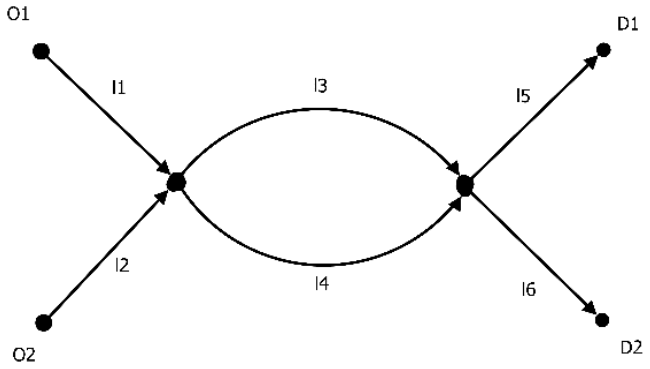


Figure 20: “Candy” network, with 6 links and 8 routes

Considering the four OD couples {O1-D1, O1-D2, O2-D1, O2-D2}, the full route set for this network can be easily enumerated, as shown in Figure 21 (thicker, red links).

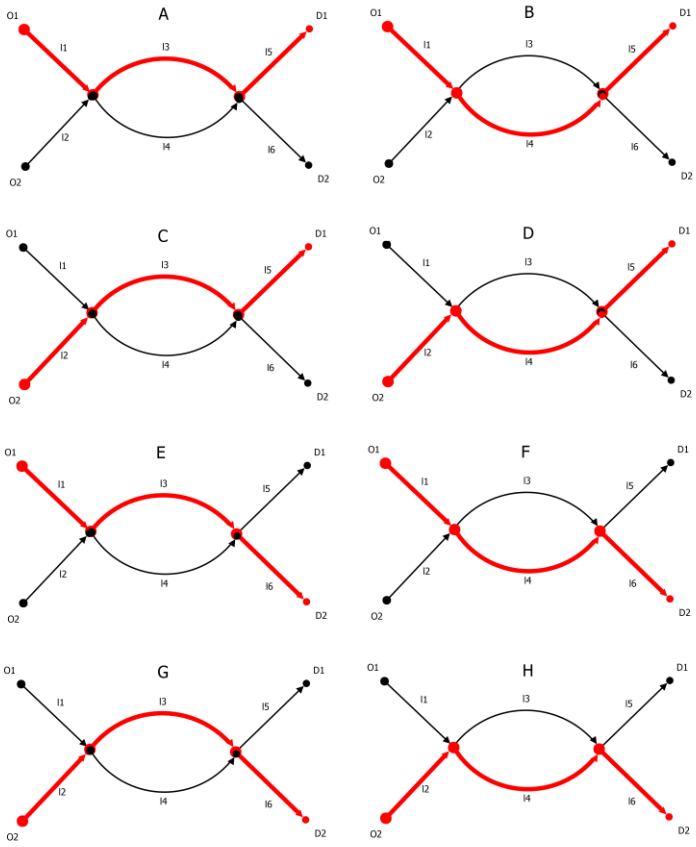


Figure 21: complete route set for the Candy network

Out of the eight routes A-H, we seek those composing the maximum independent route set. To achieve this objective, we begin by building the hypergraph $HG(V_h, L_h)$ proper to this network. As mentioned in Section 3, in the worst-case scenario the cardinalities of the vertices and arcs sets will be, respectively, $|V_h| = 256$ and $|L_h| = 32385$.

Following the same structure of Figure 3 we represent here the three instances (no culling, Algorithm 3.1 and Algorithm 3.2) in Figure 22(a-c) and the corresponding max cliques in Figure 22 (d-f), isolated from the full hypergraphs for the sake of readability. Due to the extreme density of arcs in the full hypergraph (a), links become almost undistinguishable. Table 8 summarizes the hypergraphs' and cliques' topological properties.

Table 8: summary of hypergraphs' and max cliques' topological properties and computational efforts

INSTANCE	# VERTICES	# ARCS	# VERTICES OF MAX CLIQUE	COMP. TIME [S]
FULL ENUM.	255	13731	27	1.1s
ALGORITHM 3.1	155	11935	27	0.13s
ALGORITHM 3.2	68	2113	26	0.09s

The exact enumeration and algorithm 3.1 lead to a unanimous conclusion: for this network, no combination bearing more than three routes simultaneously can in fact be chosen without violating the independence constraints (no vertex corresponding to a combination of more than two routes appears in either max cliques). In fact, no single combination of four routes out of the full set of Figure 21 exists such that all four routes, all 4 3-combinations and all 6 2-combinations of routes are independent from one another (and thus, by induction, no combination of more than four routes can respect this condition either, due to parenthood constraints).

Notice that for both Algorithms 3.1 & 3.2 the full route set has been selected as an input, simply selecting $K = 2$ for the *K*-Shortest path enumeration. Under this condition, as mentioned earlier, Algorithm 1 yields an exact solution (though discarding 100 vertices and, consequently, 1796 arcs), which can clearly be seen by comparing the two max clique solutions in Figure 22(d-e).

The result of Algorithm 3.2 is however, as can clearly be seen in Figure 22(f), approximate. By applying the more aggressive culling rule (Cul-2), some significant vertices have been removed because deemed of lower information compared to the information bounds. This results in a suboptimal solution, composed of only two routes rather than three. As shown throughout Section 4, however, this has a limited impact when dealing larger networks, and the gains in terms of computational expense are justifiable.

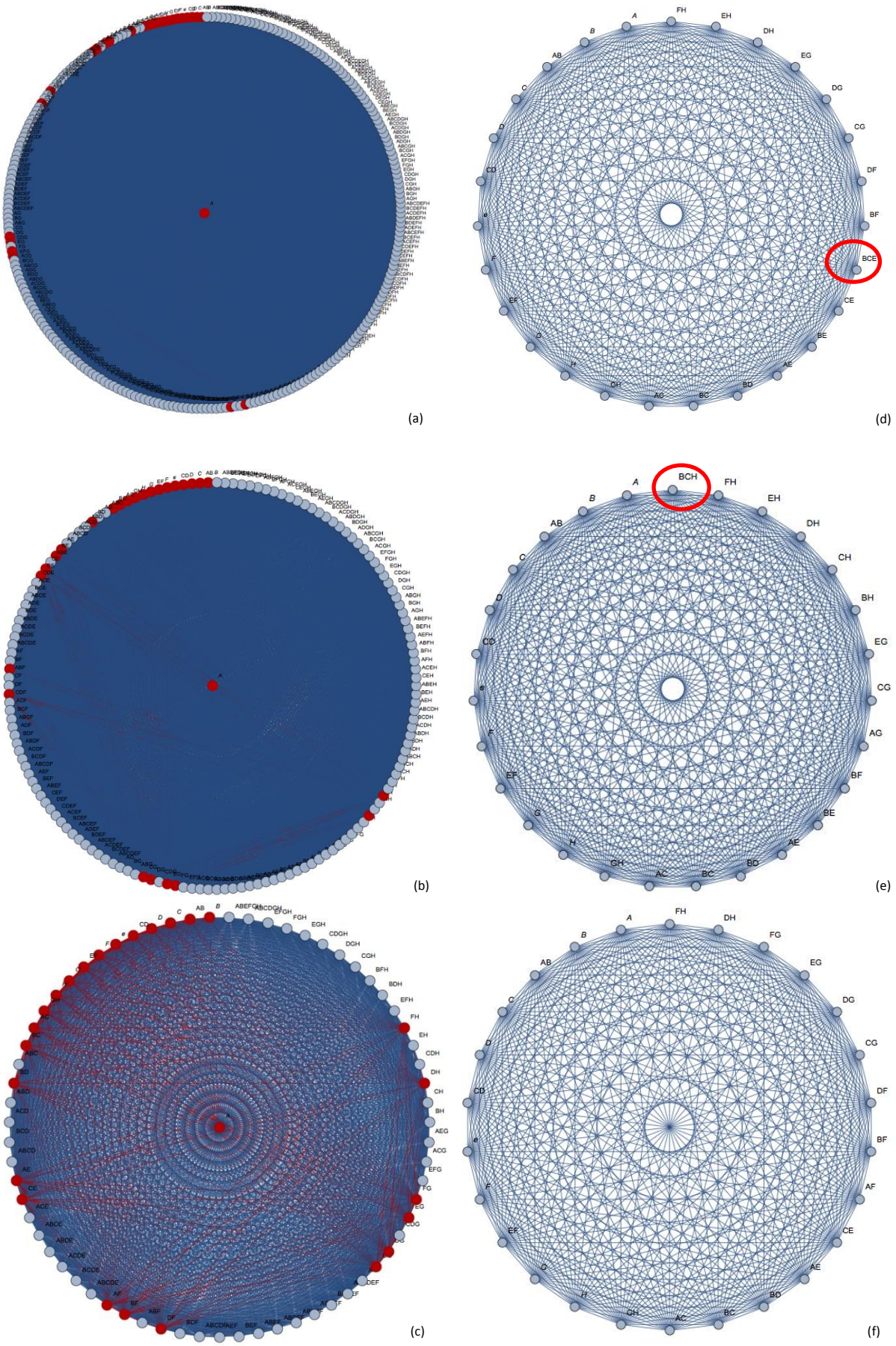
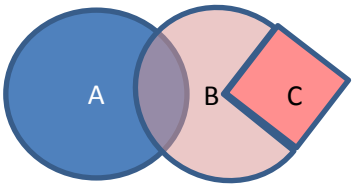


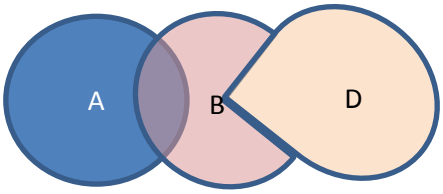
Figure 22: hypergraph representations of the Candy network for the cases of (a) no culling, (b) algorithm 1 and (c) algorithm 2, and their corresponding max cliques (d-f).

Appendix B

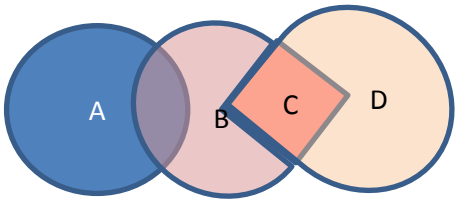
In Figure 23 we introduce a simple example showcasing the necessary but not sufficient nature of condition (22).



$$v_F = [v_A, v_B], v_M = [v_C]$$
$$v_M \not\subset v_F, |v_F| \geq |v_M| \models \mathcal{T}$$
$$\bigcup_{k=1}^{3} \binom{[v_A, v_B, v_C]}{k} \equiv K_7 \models \mathcal{T}$$



$$v_F = [v_A, v_B], v_M = [v_D]$$
$$v_M \not\subset v_F, |v_F| \geq |v_M| \models \mathcal{T}$$
$$\bigcup_{k=1}^{3} \binom{[v_A, v_B, v_D]}{k} \equiv K_7 \models \mathcal{T}$$



$$v_F = [v_A, v_B, v_C], v_M = [v_D]$$
$$v_M \not\subset v_F, |v_F| \geq |v_M| \models \mathcal{T}$$
$$\bigcup_{k=1}^{4} \binom{[v_A, v_B, v_C, v_D]}{k} \equiv K_{16} \models \mathcal{F}$$

Figure 23: Graphical interpretation of necessary condition (22).

We represent, without loss of generality, four vertices v_A, v_B, v_C, v_D pertaining to a hypergraph HG in set theoretical terms (left). Non-empty intersections between sets represent conditions the corresponding vertices exhibit common links in the bitwise representation. If the intersection between two sets includes one of the two, the included set is deemed fully dependent. Conversely, entirely disjoint sets represent vertices pertaining to the NR information category.

For the first and second scenarios, it's trivial to see that the necessary condition (22) is met: the father vertex is chosen larger than the mother vertex, and the latter clearly isn't a subset of the first. Condition (21) is also satisfied in these instances: the generated portion of the hypergraph (consisting of all k-combinations of vertices v_A, v_B, v_C for the first case and v_A, v_B, v_D for the second one) is indeed a clique.

The third case, however, depicts an instance where the necessary condition (22) holds, while the sufficient condition (21) doesn't: indeed, the combination $[v_C, v_D]$ is not independent, since $v_C \subset v_D$. As will be detailed in the algorithm specification, we deal with this issue in a post-processing phase, where vertices not pertaining to the clique are removed. As we then show in Section 6, the impact of this design choice on the overall quality of final solutions is minor, compared to the considerable gains in computational speed.

Appendix C

The full observability matrices characterizing the solutions presented in Figure 10, Ω_{HG} referring to the hypergraph approach and Ω_c for Castillo's Algorithm 1 are presented in Equation (24):

$$\Omega_{HG} = \begin{pmatrix} 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}, obs = [l_1, l_2, l_3, l_4, l_5, l_9]$$

$$\Omega_c = \begin{pmatrix} 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & -1 & 0 & -1 \end{pmatrix}, obs = [l_1, l_2, l_3, l_4, l_5, l_7, l_9, l_{11}, l_{13}] \quad (24)$$

The observed set is indeed smaller for the denser information matrix produced by the Hypergraph approach. Notably, this is accompanied by a larger Ω matrix rank (6 vs 5).

Appendix D

In this Section, the worst-case bounds on the performance of the two heuristic approaches of Algorithms 3.1 and 3.2 are discussed in terms of computational complexity and time.

The two algorithms' computational complexity will be maximum when the underlying network is such that no single vertex can be successfully culled by either. An extreme instance of such a scenario is that in which the underlying network is in fact composed by a completely disjoint set of links L. In this scenario, each link is also the one and only route between its origin and destination nodes, as exemplified in Figure 24.

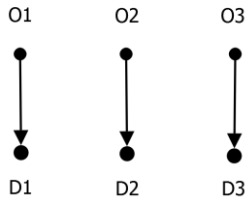


Figure 24: Fully disjoint network.

Under these conditions, it is trivial to see that no vertex can be culled by either algorithm, as indeed all routes (and combinations thereof) are by construction independent from one another, yielding a complete hypergraph $HG = K_{|R|}, R \equiv L$.

We can easily derive the computational needs of the different steps composing the hypergraph construction: both Algorithms 3.1 and 3.2 would need to perform $2^{|R|}$ operations to generate the hypergraph vertex set V_h, $\frac{2^{|R|}!}{2!(2^{|R|}-1)!}$ operations to compute the hypergraph arc set L_h and finally $2^{2|R|}$ operations to compute the parenthood constraint set matrix P_{HG}.

Under these conditions, collecting all information describing problem (14) represents the largest computational expense, while, fortunately, its solution is very trivial. Indeed, a complete hypergraph implies that the maximum clique is nothing but the hypergraph itself.

To better quantify how these computational requirements translate into actual computational times, we have performed a set of computational tests on a set of networks shaped like that of Figure 24, bearing an increasing amount of links, up to a value of $|L|=12$, beyond which memory requirements become too large to be accommodated with the available hardware (parallelization of hypergraph construction was also disabled, to ensure deterministic measurements of computational time).

In Figure 25 these results are shown, decomposed in the four basic operations listed above.

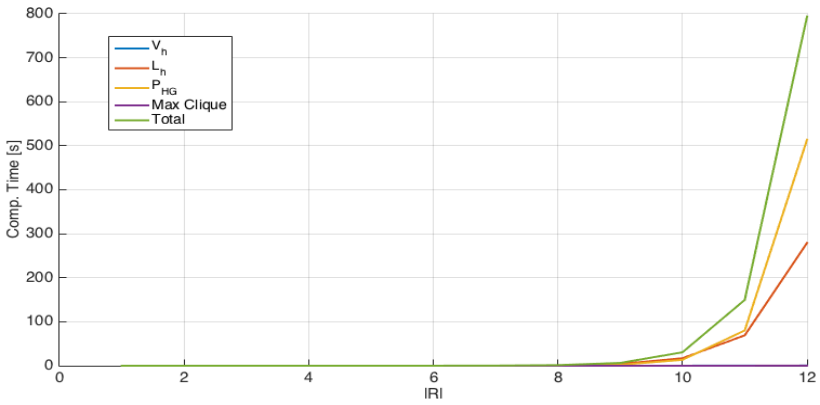


Figure 25: Measured computational times for increasing sizes of fully disjoint networks.

As can clearly be seen by analysing Figure 25, the computational time of the maximum clique is indeed very limited compared to the exponentially increasing computational times related to the hypergraph construction.

A simple estimator of the worst case computational time of either algorithm (on the tested hardware) can also be derived by fitting an exponential curve to the values collected for these tests, resulting in the following relationship:

$$Tot.\,Time \simeq 1.719 \cdot 10^{-}6 \cdot e^{1.663 \cdot |R|} s$$

Bibliography

Ahmed, A., Watling, D., Ngoduy, D., 2014. Significance of Sensor Location in Real-time Traffic State Estimation. Procedia Eng., Fourth International Symposium on Infrastructure Engineering in Developing Countries, IEDC 2013 77, 114–122. doi:10.1016/j.proeng.2014.07.012

Alidaee, B., Glover, F., Kochenberger, G., Wang, H., 2007. Solving the maximum edge weight clique problem via unconstrained quadratic programming. Eur. J. Oper. Res. 181, 592–597. doi:10.1016/j.ejor.2006.06.035

Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M., 1999. The Maximum Clique Problem, in: Du, D.-Z., Pardalos, P.M. (Eds.), Handbook of Combinatorial Optimization. Springer US, pp. 1–74. doi:10.1007/978-1-4757-3023-4_1

Cascetta, E., 2009. Transportation Systems Analysis: Models and Applications. Springer Science & Business Media.

Castillo, E., Calviño, A., Lo, H.K., Menéndez, J.M., Grande, Z., 2014. Non-planar hole-generated networks and link flow observability based on link counters. Transp. Res. Part B Methodol. 68, 239–261. doi:10.1016/j.trb.2014.06.015

Castillo, E., Cobo, A., Jubete, F., Pruneda, R., Castillo, C., 2001. An Orthogonally Based Pivoting Transformation of Matrices and Some Applications. SIAM J. Matrix Anal. Appl. 22, 666–681. doi:10.1137/S0895479898349720

Castillo, E., Conejo, A.J., Menéndez, J.M., Jimenez, P., 2008a. The observability problem in traffic network models. Comput.-Aided Civ. Infrastruct. Eng. 23, 208–222.

Castillo, E., Gallego, I., Menéndez, J.M., Jiménez, P., 2011. Link Flow Estimation in Traffic Networks on the Basis of Link Flow Observations. J. Intell. Transp. Syst. 15, 205–222. doi:10.1080/15472450.2011.620487

Castillo, E., Grande, Z., as, Calvi&#xf1, O, 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. J. Sens. 2015, e903563. doi:10.1155/2015/903563

Castillo, E., Jimenez, P., Menendez, J.M., Conejo, A.J., 2008b. The Observability Problem in Traffic Models: Algebraic and Topological Methods. IEEE Trans. Intell. Transp. Syst. 9, 275–287. doi:10.1109/TITS.2008.922929

Castillo, E., Menéndez, J.M., Sánchez-Cambronero, S., 2008c. Traffic estimation and optimal counting location without path enumeration using Bayesian networks. Comput.-Aided Civ. Infrastruct. Eng. 23, 189–207.

Cerrone, C., Cerulli, R., Gentili, M., 2015. Vehicle-ID sensor location for route flow recognition: Models and algorithms. Eur. J. Oper. Res. 247, 618–629. doi:10.1016/j.ejor.2015.05.070

Cipriani, E., Fusco, G., Gori, S., Petrelli, M., 2006. Heuristic methods for the optimal location of road traffic monitoring, in: Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE. pp. 1072–1077.

Davis, L. (ed), 1991. Handbook of genetic algorithms [WWW Document]. Van Nostrand Reinhold N. Y. URL http://papers.cumincad.org/cgi-bin/works/Show?eaca (accessed 6.12.17).

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1, 269–271. doi:10.1007/BF01386390

Fiorenzo-Catalano, S., van Nes, R., Bovy, P.H.L., 2004. CHOICE SET GENERATION FOR MULTI-MODAL TRAVEL ANALYSIS. Eur. J. Transp. Infrastruct. Res. 4.

Frejinger, E., Bierlaire, M., Ben-Akiva, M., 2009. Sampling of alternatives for route choice modeling. Transp. Res. Part B Methodol. 43, 984–994. doi:10.1016/j.trb.2009.03.001

Fu, C., Zhu, N., Ling, S., Ma, S., Huang, Y., 2016. Heterogeneous sensor location model for path reconstruction. Transp. Res. Part B Methodol. 91, 77–97. doi:10.1016/j.trb.2016.04.013

Fu, C., Zhu, N., Ma, S., 2017. A stochastic program approach for path reconstruction oriented sensor location model. Transp. Res. Part B Methodol. 102, 210–237. doi:10.1016/j.trb.2017.05.013

Gentili, M., Mirchandani, P.B., 2012. Locating sensors on traffic networks: Models, challenges and research opportunities. Transp. Res. Part C Emerg. Technol. 24, 227–255. doi:10.1016/j.trc.2012.01.004

Hadavi, M., Shafahi, Y., 2016. Vehicle identification sensor models for origin–destination estimation. Transp. Res. Part B Methodol. 89, 82–106. doi:10.1016/j.trb.2016.03.011

He, S., 2013. A graphical approach to identify sensor locations for link flow inference. Transp. Res. Part B Methodol. 51, 65–76. doi:10.1016/j.trb.2013.02.006

Hu, S.-R., Liou, H.-T., 2014. A generalized sensor location model for the estimation of network origin–destination matrices. Transp. Res. Part C Emerg. Technol. 40, 93–110. doi:10.1016/j.trc.2014.01.004

Hu, S.-R., Peeta, S., Chu, C.-H., 2009. Identification of vehicle sensor locations for link-based network traffic applications. Transp. Res. Part B Methodol. 43, 873–894. doi:10.1016/j.trb.2009.02.008

Larsson, T., Lundgren, J.T., Peterson, A., 2010. Allocation of Link Flow Detectors for Origin-Destination Matrix Estimationâ" A Comparative Study. Comput.-Aided Civ. Infrastruct. Eng. 25, 116–131. doi:10.1111/j.1467-8667.2009.00625.x

Li, X., Ouyang, Y., 2011. Reliable sensor deployment for network traffic surveillance. Transp. Res. Part B Methodol. 45, 218–231. doi:10.1016/j.trb.2010.04.005

Liu, Y., Zhu, N., Ma, S., Jia, N., 2014. Traffic sensor location approach for flow inference. IET Intell. Transp. Syst. 9, 184–192. doi:10.1049/iet-its.2014.0023

Macambira, E.M., de Souza, C.C., 2000. The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations. Eur. J. Oper. Res. 123, 346–371. doi:10.1016/S0377-2217(99)00262-3

Nachtigall, K., Voget, S., 1996. A genetic algorithm approach to periodic railway synchronization. Comput. Oper. Res. 23, 453–463. doi:10.1016/0305-0548(95)00032-1

Ng, M., 2012. Synergistic sensor location for link flow inference without path enumeration: A node-based approach. Transp. Res. Part B Methodol. 46, 781–788. doi:10.1016/j.trb.2012.02.001

Prato, C.G., 2009. Route choice modeling: past, present and future research directions. J. Choice Model. 2, 65–100. doi:10.1016/S1755-5345(13)70005-8

Putha, R., Quadrifoglio, L., Zechman, E., 2012. Comparing Ant Colony Optimization and Genetic Algorithm Approaches for Solving Traffic Signal Coordination under Oversaturation Conditions. Comput.-Aided Civ. Infrastruct. Eng. 27, 14–28. doi:10.1111/j.1467-8667.2010.00715.x

Rinaldi, M., Corman, F., Viti, F., 2015. Assessing the Effect of Route Information on Network Observability Applied to Sensor Location Problems. Transp. Res. Procedia, 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands 10, 3–12. doi:10.1016/j.trpro.2015.09.050

Rinaldi, M., Fakhraeiroudsari, F., Viti, F., Tampère, C.M.J., 2016. Improving the Accuracy of OD Estimation from Traffic Counts Employing a Partial Observability Maximizing Methodology. Presented at the Transportation Research Board 95th Annual MeetingTransportation Research Board.

S. B. Pattnaik, S. Mohan, V. M. Tom, 1998. Urban Bus Transit Route Network Design Using Genetic Algorithm. J. Transp. Eng. 124, 368–375. doi:10.1061/(ASCE)0733-947X(1998)124:4(368)

Teklu, F., Sumalee, A., Watling, D., 2007. A Genetic Algorithm Approach for Optimizing Traffic Control Signals Considering Routing. Comput.-Aided Civ. Infrastruct. Eng. 22.

Viti, F., Rinaldi, M., Corman, F., Tampère, C.M.J., 2014. Assessing partial observability in network sensor location problems. Transp. Res. Part B Methodol. 70, 65–89. doi:10.1016/j.trb.2014.08.002

Viti, F., Verbeke, W., Tampère, C., 2008. Sensor Locations for Reliable Travel Time Prediction and Dynamic Management of Traffic Networks. Transp. Res. Rec. J. Transp. Res. Board 2049, 103–110. doi:10.3141/2049-12

Xing, T., Zhou, X., Taylor, J., 2013. Designing heterogeneous sensor networks for estimating and predicting path travel time dynamics: An information-theoretic modeling approach. Transp. Res. Part B Methodol. 57, 66–90. doi:10.1016/j.trb.2013.09.007

Xu, X., Lo, H.K., Chen, A., Castillo, E., 2016. Robust network sensor location for complete link flow observability under uncertainty. Transp. Res. Part B Methodol. 88, 1–20. doi:10.1016/j.trb.2016.03.006

Yang, H., Yang, C., Gan, L., 2006. Models and algorithms for the screen line-based traffic-counting location problems. Comput. Oper. Res. 33, 836–858. doi:10.1016/j.cor.2004.08.011

Yang, H., Zhou, J., 1998. Optimal traffic counting locations for origin–destination matrix estimation. Transp. Res. Part B Methodol. 32, 109–126. doi:10.1016/S0191-2615(97)00016-7

Yen, J.Y., 1971. Finding the K Shortest Loopless Paths in a Network. Manag. Sci. 17, 712–716. doi:10.1287/mnsc.17.11.712

Zhou, X., List, G.F., 2010. An Information-Theoretic Sensor Location Model for Traffic Origin-Destination Demand Estimation Applications. Transp. Sci. 44, 254–273. doi:10.1287/trsc.1100.0319

Zhu, N., Liu, Y., Ma, S., He, Z., 2014. Mobile Traffic Sensor Routing in Dynamic Transportation Systems. IEEE Trans. Intell. Transp. Syst. 15, 2273–2285. doi:10.1109/TITS.2014.2314732