Publishing planned, live and historical public transport data on the Web with the Linked Connections framework

Julián Andrés Rojas [a], Harm Delva [a], Pieter Colpaert [a] and Ruben Verborgh [a]
[a] *IDLab, Department of Electronics and Information Systems, Ghent University-imec, Belgium*

Abstract. Exposing transport data on the Web for consumption by others poses several challenges for data publishers. In addition to planned schedules, access to live schedule updates (e.g. delays or cancellations) and historical data is fundamental to enable reliable applications and to support machine learning use cases. However publishing such dynamic data further increases the computational burden of data publishers, resulting in often unavailable historical data and live schedule updates for most public transport networks. In this paper we apply and extend the current Linked Connections approach for static data to also support cost-efficient live and historical public transport data publishing on the Web. Our contributions include (i) a reference specification and system architecture to support cost-efficient publishing of dynamic public transport schedules and historical queries; (ii) an empirical evaluation of the impact that API design aspects such as data fragmentation size, have on query evaluation performance for the route planning use case; (iii) an analysis of potential correlations of query performance with particular public transport network characteristics such as size, average degree, density, clustering coefficient and average connection duration. Results confirm that fragmentation size indeed influences route planning query performance and converge on an optimal fragment size per network, in function of its size, density and connection duration. Our approach proves to be feasible for publishing live and historical public transport data and supporting efficient route planning use cases. Yet, for bigger networks further optimizations are needed to be useful in practice. Careful design of data fragmentation strategies constitute an important factor for cost-efficient, scalable and usable publishing on the Web. Additional dataset fragmentation strategies (e.g. geospatial) may be studied for designing more scalable and performant Web APIs that adapt to particular use cases, not only limited to the public transport domain.

Keywords: Linked Data, Semantic Web, Linked Data Fragments, Linked Connections, Public Transport, Route Planning, Data Fragmentation

1. Introduction

Since it first broke onto the global stage more than 10 years ago, enabling unrestricted access to the raw data about a certain topic has been one of the guiding principles of *open data*[1]. This way, data can be freely used by anyone to address particular challenges and provide novel services [1]. Public transportation (PT) stands among the most successful domains to embrace the principles set by the open data community [2], displaying important social and economic impact [3]. Millions of people[2] around the world rely every day on open data-powered route planning applications (e.g., Google Maps, CityMapper, etc).

By definition, open data is free to be accessed and reused, but it is not free to publish open data[4], which translates in the end into restricted or even unavailable data. For the PT domain, open data have been traditionally shared through either data dumps or more complex Web APIs, both with their own merits and disadvantages in terms of cost. On the one hand, raw data

[1] https://opendatacharter.net/

[2] In 2017 Google announced having over 1 billion active users every month for Google Maps. https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users

dumps constitute a low cost data publishing strategy for data publishers, but they impose high data management costs on reusers, who need to host and maintain each dataset over which they want to offer a service. Additionally, data dumps become outdated at the moment of their creation, as they are not able to reflect any new changes on the data. On the other hand, more expressive Web APIs (usually origin–destination HTTP query interfaces) provide a low cost alternative for data reusers but limit data accessibility by imposing request limitations due to high maintenance and scalability costs [5]. Moreover, they are often designed to serve specific purposes that cannot be adjusted by client applications. Data reusers are constrained to the query capabilities and the use case supported by the API. For example, an API that calculates only the fastest routes in a PT network, may not be useful when trying to find routes that are wheelchair-friendly, or for different purposes than route planning.

These computational cost trade-offs between clients and servers (e.g, in terms of computational power, bandwidth, recency, etc) are captured by the Linked Data Fragments conceptual framework [6] and were considered for defining the Linked Connections (LC) specification [7]. LC puts forward one possible *in between* approach compared to data dumps and purpose-specific APIs, designed to model and publish PT planned schedules. By organizing vehicle departure-arrival pairs (*Connections*) into chronologically ordered and semantically enriched data documents (*fragments*), client applications can be autonomously traverse them to evaluate route planning queries [8]. In this way data publishers need only to maintain a cacheable [7] and light-weight data interface, while reusers get full flexibility over the data without the cost of maintaining the dataset.

Next to planned schedules, access to *live* schedule updates (e.g. delays or cancellations) and historical data is fundamental for building reliable user-oriented applications and supporting other use cases based on PT data, such as smart city digital twin dashboards or machine learning-based applications. These are possible only if access to live and historical data is available. However, publishing these types of data further increases the computational burden of data publishers, resulting in often unavailable historical data and live schedule updates for most PT networks.

In this paper we apply and extend the Linked Connections approach to support cost-efficient live and historical public transport data publishing on the Web. We also study how API design aspects and PT network intrinsic characteristics may influence the performance of query evaluation for the route planning use case. Our main contributions include (i) a *reference specification* and *system architecture* that foresees efficient handling of live schedule updates and allows to perform historical queries with access to precise granular data through HTTP time-based content negotiation; (ii) an *empirical study of route planning query performance* over 22 different PT networks from around the world considering different data fragmentation sizes; and (iii) a *cross-correlation of the performance results* with each network's particular size, average degree, density, clustering coefficient and average connection duration aiming on understanding how network characteristics may influence route planning query performance in practical implementations.

Results confirm that fragmentation size indeed influences route planning query performance and converges on an optimal fragment size per network. Route planning query evaluation performance is shown to be highly correlated to network size (in terms of connections) and to a lesser extent, to its density and average connection duration. Additionally, we show how knowledge of potential queries can drive a better design of data interfaces. Our approach demonstrates acceptable performance for supporting efficient route planning use cases. Yet, for larger networks further optimizations are needed to be useful in practice.

Insights on the factors that influence the performance of route planning query evaluation on LC-based applications provide a valuable asset for designing usable solutions that are fit for practical real world scenarios. For example, allowing to apply further geospatial fragmentations on top of PT networks aiming on obtaining sub networks that render higher performance for route calculations. This work stands as a contribution for the PT domain by demonstrating the feasibility of a cost-efficient approach for data sharing, and opening the door for new and innovative services and applications. It also shows how Semantic Web technologies can be applied not only to describe domain specific data, but also interfaces that enable applications to consume it, whose principles could be reused towards more generic, domain-independent and autonomous data applications.

The remainder of this paper is organized as follows. Section 2 presents an overview of related work around PT data modeling and sharing, route planning and live and historical data handling on the Web. Section 3 describes the proposed LC reference architecture. Section 4 presents the details of the empirical study on route

planning performance. Section 5 shows the obtained results. In section 6 we discuss the results and their potential correlation to PT network intrinsic characteristics. Finally on section 7 we present our conclusions and vision for future work.

2. Related Work

The field of open data has been devoted to evolving the technologies that enable to share and reuse datasets, resulting in an ecosystem of models, standards and tools. The *Linked Data principles* [9] are an example of this. Semantic Web and Linked Data technologies provide a common environment where data is given a well-defined meaning, allowing machines to interpret heterogeneous datasets by using common data models and reasoning [10, 11].

Next to the Linked Data principles for aligning datasets, we also consider the computational cost of sharing data. Different trade-offs can be observed between publishing a data dump, or providing a querying API, as described by the Linked Data Fragments conceptual framework [12]. Regarding data models and APIs for the PT domain, progress has been made as part of *Mobility-as-a-Service* (MaaS) ecosystems, aiming to provide integrated services for unified travel experiences in terms of transportation modes and payment [13].

In this section we present an overview of the main data sharing innovation efforts carried out in the PT domain, with route planning as its most prominent use case and an overview of such planning algorithms. Finally, we present related work regarding APIs to publish live and historical data on the Web.

2.1. Public transport data models

TriMet (Portland, Oregon) became the first PT operator to integrate its schedules into Google Maps in 2005. This collaboration fostered the creation of the General Transit Feed Specification[3] (GTFS), which at the time of writing, is regarded as the *de facto* standard for sharing PT data. GTFS defines the headers of 17 types of CSV files and a set of rules that describe how they relate to each other (see Figure 1). The most important files within GTFS can be listed as follows:

- **stops.txt**: Individual locations where vehicles pick up or drop off passengers.

[3]https://developers.google.com/transit/gtfs

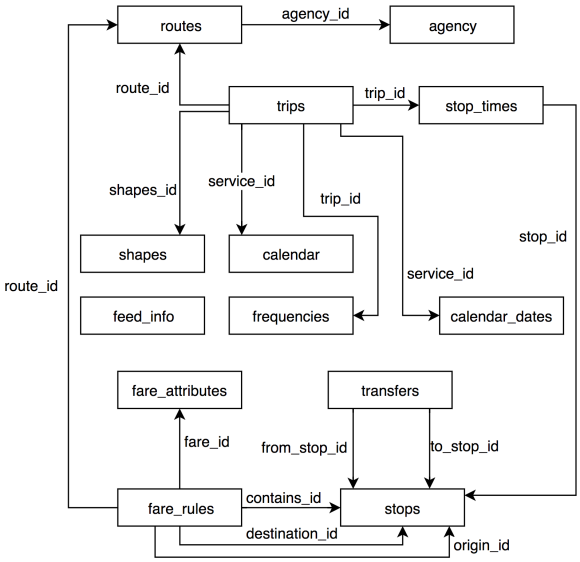

Fig. 1. The GTFS data model and its primary relations

- **routes.txt**: A route is a group of trips that are displayed to riders as a single service.
- **trips.txt**: An instantiation of a route. A trip is a sequence of two or more stops that occurs at specific time.
- **calendar.txt**: Dates for service IDs using a weekly schedule. Specify when service starts and ends, as well as days of the week where service is available.
- **stop_times.txt**: Times that a vehicle arrives at and departs from individual stops for each trip.

The European Committee for Standardization created the Transmodel[4] standard and its implementation NeTEx[5], to provide a description of conceptual models that facilitate exchanging PT network topology and timetable data, among others. NeTEx was selected by the European Union, for the provision of an EU-wide multimodal travel information service, where every member state will publish their PT-related datasets through a National Access Point (NAP). The official list of NAPs can be found online[6], however to this date only a few member states shared their data in NeTEx format, which could be attributed to the difficulty for

[4]http://www.transmodel-cen.eu/
[5]http://netex-cen.eu/
[6]https://ec.europa.eu/transport/sites/transport/files/its-national-access-points.pdf

PT operators to express their networks information in a new format and data model.

Efforts to semantically describe the different concepts, properties and relations defined by the aforementioned data models, were made for the case of GTFS with the *Linked GTFS* vocabulary[7] and for Transmodel with the Transmodel Ontology [14]. A comprehensive survey on semantic data models and vocabularies for the transport domain was performed by Katsumi et al. [15]. This survey does not focus only on PT but also includes other related aspects such parking and road traffic. The existence of so many different PT data models, sheds light on the lack of interoperability of the PT domain, but it also shows the efforts being made both from industry and public authorities to converge on well defined standards. Given it is mainly focused on modeling the concepts around PT planned schedules and that most PT data is available as such, we reuse *Linked GTFS* terminology in our approach to semantically describe PT important concepts such as stops, trips and routes. However, we take a different approach to model the granular behavior of individual trips. We pair together departure and arrival events into connections, in contrast to *Linked GTFS* that describe these events individually as a *gtfs:StopTime*. The reason for this is to facilitate the interpretation of these events to clients when evaluating route planning queries. Further details of our modeling approach are shown in section 3.

2.2. Public transport Web interfaces

Public transport data are often found on the Web as data dumps or through APIs. Static data dumps contain extensive planned schedules, which scale proportionally to the size and complexity of the transport network [16]. Most currently available dumps on the Web follow the GTFS model[8].

Public transport APIs on the other hand, can be found online spanning a wide spectrum in terms of openness, features and data structures. From the paid Google Directions[9] API, going over the *freemium* CityMapper[10] and Navitia.io[11] APIs, to the completely free and open source Open Trip Planner[12]. PT data interfaces in the wild are mostly available for route planning use cases, each with their own set of features and ad-hoc data structures. Some undergoing efforts from MaaS communities are trying to define standard API interfaces (e.g. MaaS Global[13] or TOMP[14] APIs) to harmonize data access when building MaaS applications. However, despite their heterogeneity in terms of data structures and semantics, a common pattern on their architecture design can be seen across all available APIs: the API provider's servers are responsible for handling all the computational processing burden when evaluating queries, leading in practice to feature and access-restricted APIs. We propose an alternative approach where servers only are responsible of publishing self-descriptive and departure time-sorted fragments of planned schedules, through a uniform interface and data model. This approach delegates the processing of queries to the client, in a more computational load balanced architectural setup, that ultimately lowers the costs for data publishers and brings more flexibility over the data to client applications.

2.3. Formal representation of public transport networks

Beyond the standards and interfaces used to describe and share PT data, is also important to consider the different formal representations that have been proposed to analyze and understand PT networks. Traditionally, PT networks have been defined through formalisms from graph theory and complex network science [17], with different levels of abstraction that include among others, *undirected graphs* [18, 19], *weighted and directed graphs* [20, 21], *time-expanded graphs* [22], and *time-varying graphs* [23]. Across formalisms, vertexes normally represent physical stations in the network, and edges may represent different things depending on what is intended by the topology analysis [24]. When starting from timetable (i.e., planned schedule) data, edges are usually defined to represent connections among stops. In other words, an edge is present when there is at least one vehicle that stops consecutively in two stations, when following a predetermined route [25–28].

Different metrics have been proposed to analyze and obtain insights of PT networks. General graph

[7] http://vocab.gtfs.org/gtfs.ttl

[8] GTFS dumps from around the world: https://transitfeeds.com/, https://www.transit.land/

[9] https://developers.google.com/maps/documentation/directions/overview

[10] https://citymapper.3scale.net/

[11] https://www.navitia.io/

[12] https://github.com/opentripplanner/OpenTripPlanner

[13] https://github.com/maasglobal/maas-tsp-api

[14] https://github.com/TOMP-WG/TOMP-API

theory-based metrics such as *average degree* [29], *graph density* [18], *clustering coefficient* [20], and also PT domain-specific metrics such as *average connection duration* [22] or *directness of service* [30] have been used to derive conclusions on the behavior of PT networks. For example, higher degree networks are typically associated with higher levels of network reliability [18], and higher clustering coefficients reflect higher accessibility among stations [31]. Hong et al. [32] present a compilation of studies that apply complex network metrics over different PT networks. However, even though graph metrics have been correlated to process performance in different application domains, for example to assess data quality change on dynamic knowledge graphs [33], to the best of our knowledge there are no studies that investigate potential relations of network graph characteristics and route planning query performance. We perform an evaluation in this direction and analyze how specific PT network graphs characteristics may influence route planning performance.

2.4. Route planning algorithms

Route planning is the most prominent use case over PT data and thus has been extensively studied throughout the years. Bast et al. [34] and Pajor [35] present a comparative analysis of multiple route planning algorithms over PT networks, road networks and combination thereof. Most PT algorithms are defined as extensions of Dijkstras algorithm [36] using graph-based formalizations such as *time-dependent* [37] and *time-expanded* [38] graphs to model networks. Other alternative approaches such as RAPTOR [39], CSA [40], Transfer Patterns [41] and Trip-based routing [42] exploit the basic elements of PT networks to calculate routes directly on the planned schedules.

A PT route planning query can be further specified into more concrete problems depending on the concrete use case. The literature defines different types of route planning query problems, usually defined in terms of Pareto-optimizations, that require specific algorithm implementations with varying levels of complexity [40, 43]. The simplest and most common one is the *Earliest Arrival Time* problem, where given an origin, destination and a departure time τ, an algorithm should render a journey departing no earlier than τ and arriving as soon as possible. Other common problems include the *Profile problem* variants, to calculate the set of possible journeys within a time range or the *Multi-Criteria problem* for considering additional optimization criteria over the resulting Pareto set (e.g., maximum number of transfers or transport modes).

Each algorithm requires specific data structures and indexes in order to find possible routes over PT schedules. The time-based sorted structure of our publishing approach fits the requirements for executing route planning algorithms based on the Connection Scan Algorithm (CSA). In our evaluation, we take an implementation of CSA to a client-side application and use it to evaluate *Earliest Arrival Time* queries.

2.5. Live and historical data on the Web

Live data is critical for supporting practical use cases that are useful in real scenarios. It is particularly important for PT route planning given that in practice, schedules change due to unforeseen delays and cancellations, which could render calculated routes unfeasible. GTFS-realtime[15] and SIRI[16], are among the main reference standards for live schedule updates and vehicle positions. Both define protocols to exchange live updates for planned schedules, modeled using GTFS and Transmodel standards respectively. Most currently available PT live data on the Web use the GTFS-realtime standard [5].

In the same way, historical data is fundamental for some use cases in the PT domain. Machine learning-based algorithms require training data that closely reflect reality to make predictions on a certain scenario [44]. PT operators require to perform statistical analyses based on accurate data of past events to asses the performance of their networks [45]. Unfortunately, historical PT data is not easy to come by. An example can be found through the *wayback machine* service of the Internet Archive initiative and the Belgian trains delays and disruptions site[17]. However besides not being machine readable data, this does not constitute a reliable source as snapshots are not consistently archived.

In an effort to standardize the way historical information is accessed on the Web the IETF published the RFC 7089[18], also known as the Memento framework [46]. Memento defines a protocol over HTTP to perform time-based content negotiation of Web re-

[15] https://developers.google.com/transit/gtfs-realtime
[16] http://www.transmodel-cen.eu/standards/siri/
[17] https://web.archive.org/web/20200429224623/https://www.belgiantrain.be/en/travel-info/current/ongoing-disturbances-and-works
[18] https://tools.ietf.org/html/rfc7089

sources among clients and servers. The latest version of a resource is defined as the original resource URI-R. Previous versions of URI-R are defined as Mementos URI-M $_i$ with $i = 1...n$ and can be accessed by negotiating with a *Time Gate* URI-G.

The idea of accessing and querying historical versions of data through time-based content negotiation, has been already explored by Taelman et al. for the case of time-annotated knowledge graphs [47]. In our approach we apply this principle to provide access to the history of changes of PT network schedules, which are also represented as knowledge graphs in the form of RDF. In this way we bring together both (versions of) planned and live update data, which can be queried in a cost efficient way. Design and implementation details are presented in the next section.

3. The Linked Connections framework

In previous work we introduced Linked Connections (LC)[19] as a light-weight linked open data interface for publishing PT planned schedules. It allows applications to evaluate route planning queries on the client [7, 8]. LC models PT planned schedules through departure-arrival pairs called *Connections*, which are ordered by departure time, fragmented into semantically enriched data documents and published on the Web over HTTP (see Figure 2).

Our previous work on LC mainly focused on demonstrating the feasibility of this approach and its benefits in terms of cost-efficiency for publishing PT planned schedules on the Web. We showed that indeed LC achieves a better cost-efficiency by consuming considerably less computational resources on the server-side, when compared to traditional origin-destination query interfaces for evaluating route planning queries [7]. However, we did not consider how live PT data could be managed and accessed efficiently, nor how historical data could be archived and queried. We started exploring an approach to handle live and historical data and proved its feasibility through a preliminary demonstrator [48]. Yet, a general overview of a LC-based system and a more detailed description of how its individual components could be implemented were still missing.

In this section we (i) describe the semantic specification of LC data, showing the requirements that shape the LC model. (ii) define a reference modular architecture for implementing LC-based solutions and (iii) describe in detail how we manage and provide efficient access to live and historical PT data.

[19] https://linkedconnections.org/

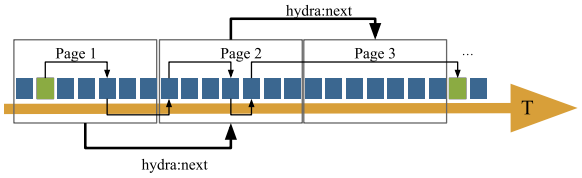

Fig. 2. Depiction of chronologically ordered linked data documents containing LC. Each document contains links to the previous and next document in the collection, that can be traversed by clients to found routes across the PT network.

3.1. Linked Connections specification

We created a specification that describes the different requirements to implement a LC data publishing interface and a set of considerations for client applications implementing route planning solutions.

LC uses *Connections* as the fundamental building block of PT data. A connection represents a vehicle going from one stop to another, at a certain point in time and without intermediary halts. In other words, a connection must contain the definition of at least a *departure stop*, a *arrival stop*, *departure time* and an *arrival time*. Additionally, a connection is related to a specific *trip* of a vehicle. This is important for client applications to interpret sets of connections as part of independent vehicle trips during route plan calculations.

We define connections as RDF graphs, following the Linked Data principles. LC data interfaces should therefore publish data, in at least one of the RDF compliant serializations (e.g., turtle, JSON-LD, N-Triples, etc). Connections are described by means of the *linked connections* ontology and also with terms from the *Linked GTFS* vocabulary. The main concepts to semantically model and represent connections are the `lc:Connection` RDF class, together with the predicates that reference departing and arrival stops and times. Table 1 describes these terms and Listing 1 shows an example of a LC using the JSON-LD serialization.

A LC data interface publishes PT network schedules as a paged collection of connections over HTTP. Each document should be served with the appropriate headers to enable both server and client-side caching. High cacheability of data is one of the biggest advantages of

Term	Description
lc:Connection	Describes a departure at a certain stop and an arrival at a different stop.
lc:CancelledConnection	Represents a previously scheduled departure and arrival that won't take place anymore.
lc:arrivalTime	The time of arrival at a certain stop. When a delay is announced, it will show that actual time of arrival.
lc:arrivalStop	A vehicle will stop here on arrival.
lc:departureTime	The time of departure at a certain stop. When a delay is announced, it will show that actual time of departure.
lc:departureStop	A vehicle will depart here.
lc:arrivalDelay	The time (in seconds) in which the lc:arrivalTime differs from the scheduled arrival time.
lc:departureDelay	The time (in seconds) in which the lc:departureTime differs from the scheduled departure time.
gtfs:trip	Indicates the specific trip to which a connection belongs to.
gtfs:pickupType	Indicates if passengers may board the vehicle at the departure stop.
gtfs:dropOffType	Indicates if passengers may get off the vehicle at the arrival stop.

Table 1

Main terms used to model and semantically define LC. The prefixes *lc* and *gtfs* stand for http://semweb.mmlab.be/ns/linkedconnections# and http://vocab.gtfs.org/gtfs.ttl# respectively.

the LC approach, in terms of cost-efficiency and scalability for data publishing interfaces. Document responses require also to enable CORS, given that data will be accessed by clients from multiple origins. Furthermore, LC defines semantically annotated hypermedia controls as part of every document's metadata. The purpose is to allow clients to discover and automatically navigate the PT schedules The hypermedia controls are defined using the Hydra vocabulary[20], including the following terms:

- **hydra:next**: Indicates the URI of the next LC document in the collection.
- **hydra:previous**: Indicates the URI previous LC document in the collection.
- **hydra:search**: Defines a URI template indicating how clients can query for a document in the collection, containing connections starting from a specific time (see Listing 2).

[20] https://www.hydra-cg.com/spec/latest/core/

```
{
    "@id": "http://example.org/IC2639/32",
    "@type": "lc:Connection",
    "lc:departureStop": {
        "@id": "http://example.org/228"
    },
    "lc:arrivalStop": {
        "@id": "http://example.org/210"
    },
    "lc:departureTime": {
        "@value": "2020-10-24T17:11:00.000Z",
        "@type": "xsd:datetime"
    },
    "lc:arrivalTime": {
        "@value": "2020-10-24T17:25:00.000Z",
        "@type": "xsd:datetime"
    },
    "gtfs:trip": {
        "@id":
        ↪  "http://example.org/IC269/20201024"
    },
    "lc:arrivalDelay": 300,
    "lc:departureDelay": 180,
    "gtfs:headsign": "Grammont",
    "gtfs:pickupType": "gtfs:Regular",
    "gtfs:dropOffType": "gtfs:Regular"
}
```

Listing 1: LC formatted in JSON-LD. The properties *departureDelay* and *arrivalDelay* indicate that live data is available for this *Connection*.

3.2. Linked Connections reference architecture

A LC system's main purpose is to publish PT schedules as a chronologically ordered collection of vehicle departures over HTTP, while taking into account live updates to the original schedules and keeping historical data available for later querying. To this end we define a reference architecture (see Figure 3) with three main modules that *generate*, *store* and *serve* LC. We also provide a complete and open-source reference implementation of this architecture as a Node.js application[21].

LC Generator This module is responsible for creating LC. It takes GTFS (planned schedules) and GTFS-realtime (live updates) data sources as input, given that most PT data is available in these formats. We provide implementations for both modules through the *gtfs2lc*[22] and *gtfsrt2lc*[23] Node.js libraries. However,

[21] https://github.com/linkedconnections/linked-connections-server

[22] https://github.com/linkedconnections/gtfs2lc

[23] https://github.com/linkedconnections/gtfsrt2lc

```
{
    "hydra:search": {
        "@type": "hydra:IriTemplate",
        "hydra:template":
        ↪ "http://example.org/connections{
        ↪ {?departureTime}",
        "hydra:variableRepresentation":
        ↪ "hydra:BasicRepresentation",
        "hydra:mapping": {
            "@type": "IriTemplateMapping",
            "hydra:variable":
            ↪ "departureTime",
            "hydra:required": true
        }
    }
}
```

Listing 2: Hydra search form defining a URI template for accessing LC documents with connections departing no earlier than the requested time. It explicitly defines how clients can request specific documents and the variables they are allowed to use. In this case the only variable is the `departureTime`.

thanks to the modular nature of the architecture, it is possible to replace these modules with any other interfaces capable of creating LC from different data sources (e.g., Transmodel, APIs, etc). One of the most important aspects that need to be considered when creating LC is the provision of a stable identification (URI) strategy that remains valid across versions of the data sources. We make possible to define such strategy using URI templates as defined by the RFC 6570 specification[24].

Data Storage The output of *LC Generator* is received by this module, which proceeds to fragment and store the data according to a given fragment size. Static LC (i.e. data coming from a planned schedules) are stored as individual files that correspond to the pages of the time-ordered LC collection. Additionally, files containing the set of *stops* and *routes* of the PT network are kept to be served as static documents too, since they are usually needed by route planning applications. Live updates are also stored as files following a log-like approach, where delays, ahead of time and cancellation reports for every single connection are written down. Files in both cases are named using the first departure datetime they contain to facilitate later connection lookups.

[24]https://tools.ietf.org/html/rfc6570

Web Server This module defines the interfaces through which LC and other related PT data may be accessed by client applications. The HTTP interfaces supported by the LC Web server are as follows:

- `/connections`: This interface provides access to the LC documents. It receives a departure time query parameter, as seen in Listing 2, used to obtain the document with connections departing on a specific time. If not provided it will resolve to the current time document.
- `/stops`: This interface returns the complete set of stops defined for the PT network as a static document. Stops are described with terms from the *Linked GTFS* vocabulary.
- `/routes`: This interface returns the complete set of routes available in the PT network as a static document. It includes information like route number/name, color or type of vehicle (e.g. metro, tram, bus, etc) which are also described with the *Linked GTFS* vocabulary. Route data are useful for displaying route plan results in user applications.
- `/catalog`: This interface provides a catalog definition given using the DCAT vocabulary. It describes the different data sources published on the server, including their access URLs, supported media type formats, last issued date, license information, among other metadata. Its main purpose is to increase discoverability of the data.

The *Web Server* module also contains submodules responsible for resolving LC documents requests in an efficient way. Particularly the architecture defines three specific submodules for supporting requests that include live data, historical data and also static data. The *live data manager* submodule takes care of serving LC documents that include the latest connection updates. Details of its internal structure and functionality are presented later in section 3.3. In the same way, the *historical data manager* handles serving previous versions of LC documents through HTTP time-based content negotiation using the Memento protocol. The details of how this module works are described in section 3.4. Lastly, the *static data manager* handles requests for static resources, namely *stops*, *routes* and the server's DCAT metadata.

time-based content

3.3. Serving live Linked Connections

Managing and serving live schedules updates, without sacrificing the cost-efficiency of the data publish-

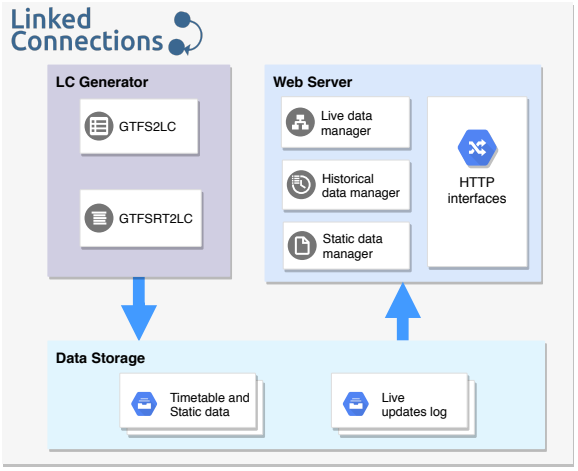

Fig. 3. Reference architecture for LC-based systems.

ing interface, constitutes one of our main contributions in this paper. In previous work we studied pushing (server-sent events) and polling (HTTP) interfaces to exchange live PT data with client applications and keep route planning results updated in a cost-efficient way. We found that a polling approach consumes less resources on the data publishing side and clients only experience a slightly higher bandwidth consumption, compared to a pushing approach [49]. However, our implementation for serving live LC was done in a naive way. We merged scheduled LC documents with their latest updates on request time, with significant negative impact on response times.

We introduce a more elaborated approach to reduce response times of LC document requests without compromising cost-efficiency. The set of time-sorted Linked Connections $C = [c_0, c_1, \ldots, c_n]$ where dt_k is the departure time of c_k and $dt_k \leqslant dt_{k+1}$, can be modeled as an AVL tree [50]. AVL trees are self-balancing binary search trees, where at any time, the height difference between two child subtrees of any node is not bigger than 1. Insert and delete operations are performed in logarithmic time and the strict balancing ensures consistent response times on data lookups. We implemented an AVL tree in our LC architecture represented by the *live data manager* submodule in Figure 3. The tree creates a time window view over the LC collection, spanning from the current time until a configurable time in the future. This time window is periodically adjusted by shifting forward in time, based on the assumption that most route planning queries will request future routes and also to avoid unnecessary memory consumption by keeping old connections. However, data outside the time window can still be provided by merging scheduled documents and their updates on request time. The AVL tree data structure is updated accordingly (i.e. adding, removing and reorganizing connections) upon reception of schedule change reports. This allows for fast LC document responses containing the latest schedules updates. Figure 4 shows an example of an AVL tree of LC and how the tree is adjusted when a connection is reported to have departure delay.

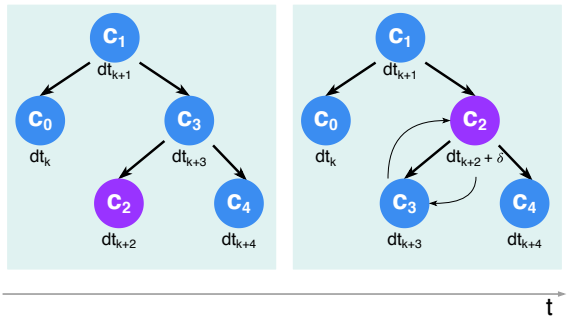

Fig. 4. LC AVL tree for updated departure schedules. In this example c_2's departure time is increased by a delay δ, triggering a tree reorganization to maintain the chronological ordering of the collection.

The AVL tree is initially generated by scanning over the scheduled LC, kept by the *Data Storage* module on server boot time. Once created, the live update logs are constantly monitored and trigger tree reorganizations when new reports are received.

3.4. Serving historical Linked Connections

Another important contribution of this paper, is providing the ability for serving historical LC data. We allow querying not only for past planned schedules but also for historical live data reports. This means it is possible to obtain the actual vehicle departures as they were reported at different points in time. For example, we could request for the departures of yesterday at 08:00h as they were expected to be yesterday at 07:00h and also later at 07:50h, seeing possibly that a connection that was on time at 07:00h was later reported to be delayed at 07:50h. In this way is possible to reproduce the stream of events of a PT network at a granularity given by the frequency of live update reports. Access to this data could support analytical studies to better understand the behavior of PT networks and also how they could be improved.

```
GET /connections?departureTime=2020-10-
↪ 15T08:00:00.000Z
↪ HTTP/1.1
Host: example.org
Accept-Datetime: Thu, 15 Oct 2020 07:35:00
↪ GMT
Connection: close
```

Listing 3: Hydra URI template for accessing LC documents containing connections with departure times equal or bigger than the requested time.

We make this possible through the HTTP Memento protocol. Given the document-based nature of LC, it is possible to request past versions of a specific document, as it was at a certain point in time. Memento defines different patterns to perform time-based content negotiation. We implemented pattern 1.1 where URI-R = URI-G and 302-style negotiation is performed. This means that the original resource acts as its own time gate and clients receive a 302 HTTP response containing a `Location` header with the URI of the Memento. An example GET request is shown in Listing 3, asking for connections departing from 08:00h as they were reported at 07:35h. The specific version time is given through the `Accept-Datetime` header, as defined by the Memento protocol.

Performing these kind of queries is possible thanks to the way LC are stored in the *Data Storage* module, as separate sets for both scheduled and live update data. When a Memento request is received, the system gets first the LC fragment containing the originally scheduled connections. This first step may seem trivial but is necessary to consider that there may be multiple versions of overlapping planned schedules. Therefore, the system needs to select the version issued closest to the specified `Accept-Datetime` date, before integrating live reports. Then the system goes over the live update logs for this specific LC document, retrieving and merging all the updates received up until the `Accept-Datetime` date. As mentioned in section 3.3 this could be considered as a naive approach which may increase response times. However we part from the assumption that historical data queries are not as performance-critical as live data queries for route planning purposes, and can still be resolved within reasonable time following this approach.

3.5. Linked Connections client

The chronological ordered collection of connections defined by a LC system, is a fitting data structure to perform the *Connection Scan Algorithm* (CSA), proposed by Dibbelt et al. [40]. Given a departure stop, arrival stop and departure time, CSA will go over the collection of connections, progressively building a minimum spanning tree of reachable destinations. The algorithm performs this process until it reaches the desired arrival stop, rendering in this way, the earliest arrival journey possible (if any). This provides a solution for the *Earliest Arrival Time* (EAT) problem. In the case of LC, a client performing the CSA algorithm can scan through the collection of connections by downloading LC documents and following the defined hypermedia controls to traverse it. We provide an implementation of CSA on the Planner.js JavaScript library[25], which can be used both on server (Node.js) and client-side applications.

4. Evaluation

To support efficient PT data publishing and real-world practical use cases such as route planning, it is fundamental to achieve high performance for query processing. Therefore, we need to understand the factors that influence performance and the API design aspects that could be adjusted to optimize them. One of the aspects that can be controlled on LC systems, is the LC data fragment (document) size, in terms of maximum number of connections they can contain. In previous work, we established arbitrary time window ranges (e.g. 10 minutes) per document, aiming to have LC documents of reasonable size to be transmitted to clients over HTTP. However in practice, this resulted in a wide range of sizes for LC documents, which in turn translated to unpredictable query evaluation performance. This is due to PT networks normally exhibiting significantly higher numbers of connections during peak hours, which also increase proportionally to the number of trips that take place on the network. For this reason we opted for establishing a (configurable) fixed size for LC documents, given by a maximum number of connections allowed per document, that gives more stable document response times and thus more predictable route planning performance. Determining the size of LC documents takes us to our first research question and hypothesis:

- **RQ1**: What is the optimal data fragment size for maximizing route planning query performance of a PT network modeled and published as Linked Connections?

[25] https://planner.js.org/

– **H1**: There is an optimal LC data fragment size for PT networks that renders the highest route planning evaluation performance.

This hypothesis comes from considering that are too big fragments will increase response times and processing effort for individual requests, and fragments that are too small will require more HTTP request-response cycles, both cases resulting is poorer query evaluation performance. Therefore, finding the optimal LC document size (max number of connections per document) of individual PT networks is an important design aspect for LC systems but it does not provide a complete picture of the principles that guide better query performance. Finding a generalized solution that maximizes query performance when publishing LC, requires determining the patterns present when high performance is achieved. For this we need to observe the properties of the PT networks themselves, aiming on finding the ideal conditions for maximum route planning performance. This takes us to our second research question and hypothesis:

– **RQ2**: What correlations exist between route planning query performance over LC-based data interfaces and the topological properties of PT networks graphs?
– **H2**: There is a set of topological properties of PT networks that are correlated with better route planning query performance over LC. That is, it is possible to find PT networks whose topological characteristics improve route planning query performance when published over LC interfaces.

To tackle these research questions, we performed an empirical evaluation using 22 real-world PT networks, where we fragmented their corresponding LC collections and measured the performance of route planning queries with different fragments sizes. Additionally, we measured particular network graph properties such as size, average degree, density, clustering coefficient, and also a PT specific characteristic such as the average connection duration. We ran this experiment on two machines acting as server and client, both with a 2x Quad core Intel E5520 (2.2GHz) CPU and 12GB of RAM. Both machines were set in a local network to avoid network latencies affecting results. For reproducibility, we made available the original data sources, query sets, tools and obtained results of this evaluation[26].

[26] https://github.com/julianrojas87/lc-evaluation-swj

4.1. Fragmenting Linked Connections

For measuring route planning query performance on the 22 PT networks selected for this evaluation, we took the following steps.

Generate Linked Connections We converted the PT networks to LC from GTFS data sources found on the Web as open data[27]. For this we used the *gtfs2lc* Node.js library.

Busiest day We looked for the busiest day of every PT network, by counting the number of connections present each day. The busiest day acts as a representative subset of the planned schedules, given that for any other day, route planning algorithms will need to process less data to answer queries. We also make the assumption that in practical scenarios, most PT route planning queries will be normally evaluated within the span of one day.

Smallest fragment possible Fragmentation of LC collections is driven by a configurable maximum number of connections allowed per document. However, a fragmentation cannot be arbitrarily small, because PT networks in LC systems have a lower bound of connections per document. This lower bound is determined by the maximum number of simultaneous connections in the smallest time interval possible in the schedules. In other words, connections sharing the same exact departure time, cannot be fragmented across different LC documents as this would break the indexing mechanisms that LC systems rely on. All of the 22 evaluated PT networks provide departure times with a resolution of one minute, therefore, we could determine the smallest possible fragment, by looking for the busiest minute, i.e., the maximum number of simultaneous connections in one minute. With this lower bound we were able to fragment the rest of the collection in fragments containing similar number of connections and hence a similar size, without breaking the time-based index.

Fragmentation sets Knowing the lower bound, we proceeded to fragment the LC collections starting from their lower bound and progressively increasing the number of connections per fragment. We used fixed sizes of 10, 50, 100, 300, 500, 1,000, 3,000, 5,000, 10,000, 20,000 and 30,000 connections per fragment; since smaller PT networks have a low total number of

[27] https://www.transit.land/feeds

connections, we stopped fragmenting them when the fragment size reached the size of the entire collection. We used these fragmentation sets of each PT network to measure and compare route planning query performance.

4.2. Route planning queries

Performance of route planning query evaluation depends not only on how the data is structured and organized but also on the type of queries that need to be processed. The literature defines different classes of problems for the route planning use case that involve a varying number of variables. To minimize the number of variables that may influence our performance measures, we selected the most simple type of problem, namely the *Earliest Arrival Time* problem. The goal in this case, is to find the journey with the shortest travel duration between origin and destination, given a minimum departure time. Processing of these queries focuses only on optimizing the arrival time, disregarding other common variables such as maximum number of transfers or transportation modes. In our test case, transfers are possible but constrained only to a maximum walking time of 10 minutes at an average speed of 3km/h (500 meters). For simplicity, calculation of transfer stops within walking distance are performed considering only the euclidean distance between stops. EAT queries can be processed easily over LC interfaces via the CSA algorithm. The algorithm can perform a single scan over the LC collection until it finds a complete journey (if any), which is guaranteed to be the earliest arrival thanks to the chronological ordering of the LC collection.

Query selection for each PT network in our evaluation was performed at random, for origin-destination stop pairs at any time of the day. We randomly generated 100 (solvable) queries for each network and replayed them against each fragmentation setup, measuring the time needed for each query to be evaluated. We also counted the number of connections that CSA had to process to evaluate each individual query and called this the number of Scanned Connections needed by the Query (SCQ). $E(\text{SCQ})$ then denotes the expected value, or the average, of the number of connections for the query set. Its standard deviation is denoted by σ. Both metrics not only provide insights on the query set, but also on the network itself. For example, lower number of scanned connections per query could mean the presence of shorter EAT route queries, i.e., CSA needs to scan only a few connections to evaluate the queries.

Query set	$E(\text{SCQ})$	σ	Distribution
Kobe-Subway	140	150	
Netherlands-Waterbus	120	80	
San Francisco-BART	370	180	
Thailand-Greenbus	450	220	
Auckland-Waiheke	390	740	
Sydney-Trainlink	380	190	
London-Tube	13,030	7,850	
Germany-DB	2,970	1,420	
Belgium-NMBS	9,760	6,000	
Spain-RENFE	4,100	1,420	
Amsterdam-GVB	9,190	10,660	
EU-Flixbus	33,470	9,280	
New Zealand-Bus	20,510	22,480	
Brussels-STIB	13,770	9,430	
Nairobi-SACCO	5,070	1,150	
New York-MTABC	28,220	24,250	
France-SNCF	39,540	15,090	
Madrid-CRTM	87,420	45,450	
Helsinki-HSL	76,440	68,350	
Chicago-CTA	64,240	42,290	
Flanders-De Lijn	322,240	141,690	
Wallonia-TEC	234,310	109,710	

Table 2

Average number of connections scanned to process the randomized query set of each PT network.

It could be also an indication of fewer simultaneous trips in the network. Thus, CSA is able to find a route without having to scan many connections from other irrelevant trips that happen at the same time.

Table 2 shows a summary of $E(\text{SCQ})$, σ and a visualization of the SCQ distribution in the form of a sparkline for each query set. The values of $E(\text{SCQ})$ show that the lowest value belongs to *Netherlands-Waterbus* with an average of 120 connections, and the highest to *Flanders-De Lijn* with an average of 322,000 connections. Later in section 5 we show that these values are aligned with *Netherlands-Waterbus* being among the smallest and *Flanders-De Lijn* being among the biggest networks in terms of both stops and connections. σ and the distribution reflect the random nature of the query sets. We see that the most balanced ones are those whose σ is closer to $E(\text{SCQ})/2$.

4.3. Public Transport Network and metrics

Considering that PT networks topologies are inherently time-dependent, we opted to model them as *Time-Varying Graphs* (*TVG*). The main purpose was to capture more accurately their dynamic behavior and

evolution [23]. Traditional aggregated static graphs may be a severe oversimplification that fails to represent the number and particularly the frequency of relations that take place in a dynamic system [51]. As an example of how much a PT network topology may change over time, Figure 5 shows 4 snapshots of the Belgian train PT network graph, taken at different points in time.

*TVG*s are typically defined by an ordered-set of T snapshot graphs $G_1, G_2, \ldots, G_T$, where each G_t represents a state of the network at a certain point in time. $G_t = (V, E_t)$ where V is the constant set of vertexes (stops) and E_t represents the temporal configuration of edges (connections) that take place on the network at t. We take T at the maximum resolution allowed by the timetable data of 1 minute, to capture better the state of the networks during the observed time interval. Edges may be persistent across graph snapshots, according to the travel duration of the connections they represent.

Based on related work on analytical frameworks to study PT networks [18–21, 23], but mainly aiming to reflect their dynamic behavior and topological changes over time, we decided to observe the following graph properties of each network:

- *Size*: Size is a basic graph property, in this case interpreted as the total number of stops $|V|$ present on the network.
- *Average Degree*: Degree k is measured on a vertex as the sum of its incoming and outgoing edges, interpreted in this case as departing and arriving connections. For every graph snapshot Gt, we take the average degree of all vertexes. The *TVG* average Degree is then calculated as the average graph Degree over graph snapshots:

$$K = \frac{1}{|T| * |V|} \sum_{t \in T} \sum_{v \in V} k$$

  The average degree of a network shows how connected is each vertex in the network [32].
- *Density*: Graph Density D is an indicator aimed at measuring how close is the network structure to a complete graph. It is defined as the ratio of existing edges and the total number of possible edges in the network. We calculated the total Density of the *TVG* as the average Density of the individual graph snapshots:

$$D = \frac{1}{|T|} \sum_{t \in T} \frac{|E_t|}{|V| * (|V| - 1)}$$

  An increased density is usually an indication of reduced time travelling in PT networks [18].
- *Clustering Coefficient*: Clustering Coefficient C is a measurement of how well connected are the neighbors of a given vertex. Is defined as the ratio of existing edges and total possible edges among neighbors of a vertex, which is averaged for all the vertexes in the network. We measured the total C of the *TVG* as the average for all the snapshot graphs Gt:

$$C = \frac{1}{|T| * |V|} \sum_{t \in T} \sum_{v \in V} \frac{2|e|}{|n| * (|n| - 1)}$$

  where e is the number of edges present among neighbors of vertex v and n is the total number of neighbors of v. A highly clustered network is usually a reflection of a better connected and accessible network [31].
- *Average Connection Duration*: This metric is a particular measure of time-dependent networks, which indicates in this case, how long are the trips that occur on the network [22]. From a LC system perspective is interesting to see how longer or shorter trips in PT network may influence route planning performance, considering the time-based nature of LC data interfaces. We calculate Average Connection Duration over the LC collection as the average difference of arrival and departure times for every connection $ACD = c_{at} - c_{dt}$.

5. Results

In this section we present the measurements obtained during our evaluation process. To provide a contextualized view of the results, we first summarize the (network graph) characteristic of each PT network considered in the evaluation. Then we present the results or route planning query evaluation performance using different fragmentation sizes. Afterwards, we zoom in into networks that show high performance to highlight their particular behavior. Lastly we contrast each of the considered metrics against the query performance results to visualize potential correlations.

5.1. Evaluated networks and metric values

Table 3 presents a condensed view of each of the 22 PT networks considered during this evaluation. Aim-

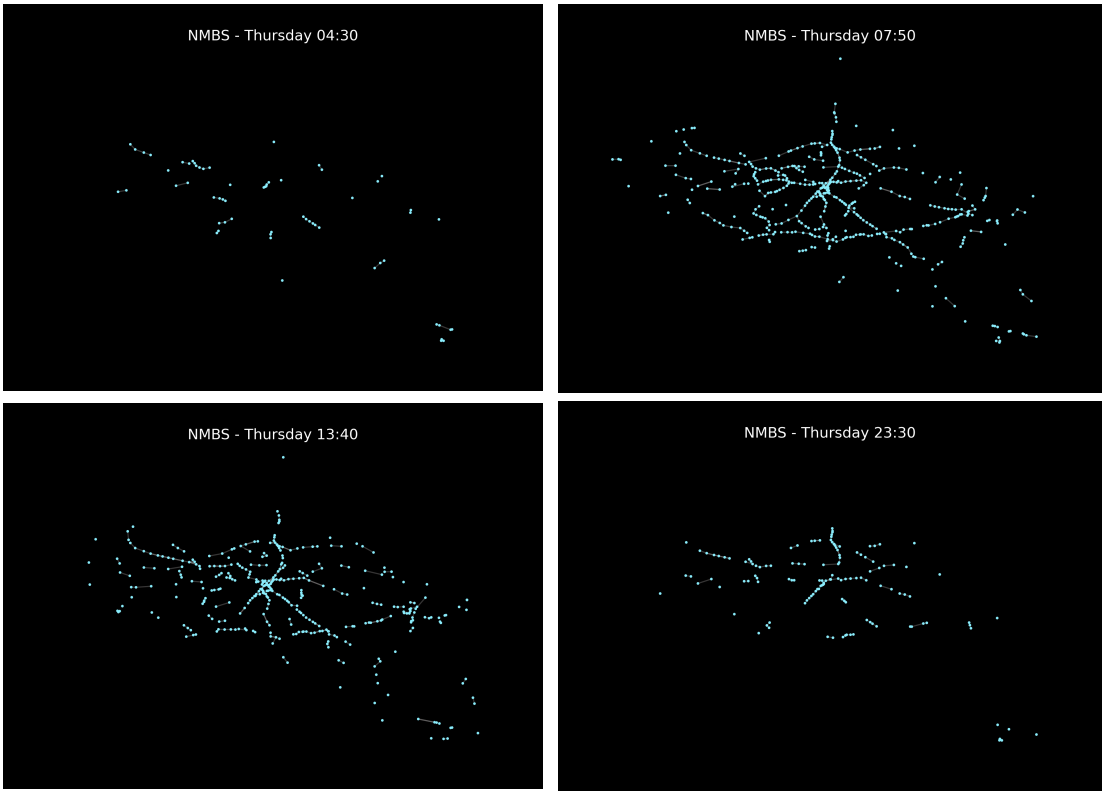

Fig. 5. Network graph snapshot of the Belgian train PT operator NMBS, taken over the busiest day of their timetable.

ing on getting generalizable results, we selected a representative set of real and heterogeneous PT networks from around the world. Besides varying on network graph property values, we also chose networks of different types, in terms of modes of transport and geographical coverage (urban, regional, national and international).

We observe high heterogeneity in the different measured metrics. For the total number of stops ($|V|$), we have the *Kobe-Subway* network as the smallest with 27 stops, and the *Wallonia-TEC* network as the biggest with a total of 31,131 stops. In the case of total number of trips, *Sydney-Trainlink* has the least number with 103 and *Flanders-De Lijn* has the highest number with 33,959. *Sydney-Trainlink* has also the lowest number of connections with 891 and *Chicago-CTA* has the highest with almost 1.13 million connections.

We can see that more stops does not necessarily means more connections. *Sydney-Trainlink* (least connections) has 13 times more stops than *Kobe-Subway* (least stops). In the same way, *Chicago-CTA* (most connections) has less than half the stops of *Wallonia-TEC* (most stops). Having the least connections is a reflection of also having the least trips in the case of *Sydney-Trainlink*. However, *Flanders-De Lijn* (most trips) has 5.6 times more trips but 30% less connections than *Chicago-CTA* (most connections). Such difference is explained by *Chicago-CTA*'s trips being larger in terms of visited stops, which translates into higher number of connections.

The smallest fragment size, which is given in number of connections, has *Auckland-Waiheke* as the network with the smallest fragment possible: 5 connections per fragment. *Flanders-De Lijn* has the biggest among all networks with a minimum possible fragment of 1.8k connections. This metric reflects how many simultaneous connections take place at the busiest moment of the schedule.

Looking at the average degree K, *Auckland-Waiheke* shows again the lowest value with 0.15 and *london-tube* presents the highest with 1056.55, showing a significant difference compared to the rest of the networks. This indicates that throughout the day, most of *London-Tube*'s stops are constantly active, which is evident by the high number of connections compared to the low total number of stops showed by this

PT network	stops	trips	connections	smallest fragment	K	$D * 1000$	C	ACD
Kobe-Subway	27	617	6,086	16	6.59	84.55	0	2.57
Netherlands-Waterbus	44	515	936	7	1.17	9.14	0	11.08
San Francisco-BART	50	754	7,755	15	9.28	63.14	0.19	4.53
Thailand-Greenbus	112	137	1,024	16	3.20	9.62	0.97	83.81
Auckland-Waiheke	125	243	6,020	5	0.15	0.41	0	1.60
Sydney-Trainlink	361	103	891	7	0.19	0.17	0.01	51.24
London-Tube	379	15,356	321,952	376	1,056.55	931.70	6.77	2.51
Germany-DB	433	677	7,680	21	4.47	3.45	6.58	33.05
Belgium-NMBS	606	4,556	57,950	94	35.36	19.48	0.54	5.66
Spain-RENFE	714	997	6,159	27	3.48	1.62	2.69	32.97
Amsterdam-GVB	1,356	11,367	180,695	71	0.68	0.24	0.001	2.11
EU-Flixbus	1,744	8,726	51,636	386	162.01	30.98	27.14	133.05
New Zealand-Bus	2,259	4,678	153,690	59	0.50	0.07	0.03	2.01
Brussels-STIB	2,316	19,557	350,038	189	2.47	0.35	0.13	1.76
Nairobi-SACCO	2,787	264	5,855	259	6.71	0.80	1.26	4.69
New York-MTABC	3,590	11,028	343,582	130	1.19	0.11	0.59	4.19
France-SNCF	4,646	10,541	79,796	180	20.19	1.44	2.57	16.52
Madrid-CRTM	5,192	27,538	706,642	247	3.93	0.25	2.61	5.49
Helsinki-HSL	8,155	25,887	689,834	877	130.76	5.34	3.78	1.62
Chicago-CTA	11,042	20,058	1,128,828	164	2.20	0.06	0.33	1.37
Flanders-De Lijn	29,905	33,959	826,572	1861	117.11	1.30	1.62	1.56
Wallonia-TEC	31,131	21,062	623,808	1207	36.02	0.38	3.99	1.55

Table 3

Set of evaluated PT networks and their metric values. The networks are organized from the smallest to the biggest with respect to the number of active stops during their busiest day. Number of trips and connections correspond to the total amount that took place during the busiest day of the schedule. *K* is the average degree, *D* is the density (shown as a factor of 1000 to facilitate readability), *C* is the clustering coefficient, *ACD* is the average connection duration (in minutes).

network. For density D, we observe that values range from 0.00006 for *chicago-cta* to 0.93 for *London-Tube*. We also see that networks with high K and relatively lower number of stops show the highest values of D, as is the case of *Kobe-Subway*, *San Francisco-BART* and *London-Tube*.

For clustering coefficient C, we can see that three of the networks, namely *Auckland-Waiheke*, *Netherlands-Waterbus* and *Kobe-Subway* have $C = 0$. We see that these networks have in common a relatively small number of stops, a low number of simultaneous connections (given by the smallest possible fragment size) and relatively low *ACD*. In contrast to *EU-Flixbus* that has the highest $C = 27.14$ and also the highest *ACD*. This pattern can be explained by the fact that having low number of stops and *ACD*, lowers the probability to find a stop v_k that at any given time, has connections with two neighbor stops v_{k+1} and v_{k+2}, at the same time that v_{k+1} is also connected to v_{k+2}. In the case of *EU-Flixbus* we could infer that is easier to find simultaneous busses travelling among neighbor stops, given the higher *ACD* of this network. An example of this scenario in *EU-Flixbus* is show in Listing 4.

```
Paris CDG Airport @00:10
    ---> Brussels South @03:30

Paris CDG Airport @00:20
    ---> Paris (Bercy Seine) @00:55

Paris (Bercy Seine) @00:53
    ---> Brussels South @03:30
```

Listing 4: Example of a cluster in *EU-Flixbus*. Given the long duration of the two connections departing from France to *Brussels South*, when the connection between the two french stops takes place, the other two connections are still happening, therefore a cluster (triangle) can be formed in the graph.

Lastly, we see that the values for average connection duration range from 1.37 minutes of *Chicago-CTA* to 133.05 minutes of *EU-Flixbus*. This is expected, since urban networks normally have shorter connection du-

rations compared to nation-wide or international networks such as *Thailand-Greenbus* and *EU-Flixbus*.

5.2. Route planning performance vs fragmentation

Figure 6 presents an overview of the results obtained from the route planning performance evaluation, over different sets of LC data fragmentation. We use a per-connection result to remove the influence of route query length. Unevenly distributed query sets with too many long or short routes, influence the overall results of response time for a given fragmentation. This is due to CSA having a time complexity of $O(n)$ with n being the total number of connections processed to calculate an EAT route. For this reason, we used a normalized value of response time per connection, obtained by dividing the measured response time of a query by the number of connections processed to evaluate it. We thus study the time needed to process one connection per route planning query.

The top left plot in figure 6, shows the results for the three smallest networks in terms of total connections (< 1,100). Fragmentation for these networks were only possible until 500 connections/fragment for *Netherlands-Waterbus* and *Sydney-Trainlink*, and until 1,000 connections/fragment for *Thailand-Greenbus*. Bigger fragmentation for these networks would mean that the entire collection of connections would fit in only one fragment. All three networks render their best response time per connection with a fragmentation of 100 connections per fragment (LC document). *Netherlands-Waterbus* shows worse response time per connection compared to both *Sydney-Trainlink* and *Thailand-Greenbus*, which can be explained by the higher number of connections per query that these network require to be processed (table 2). We see similar trends for the three networks, converging on 100 connections per fragment and showing degraded performance with either smaller or bigger fragmentation.

The top right plot in figure 6, brings together 6 different networks with total amounts of connections ranging between 5,000 and 8,000. Results are more varied compared to plot A, both in terms of optimal fragmentation and response time per connection. Optimal fragmentation values are found mostly on 1,000 or less with low variation. Only *Nairobi-SACCO* shows a higher value of 3,000 connections/fragment. *Auckland-Waiheke* and *San Francisco-BART* performed best both with 100 connections/fragment, yet they show a significant difference in terms of response time per connection, with 3.18 and 0.23 ms/connection respectively. Referring to table 3, we can see that both networks have relatively similar characteristics on almost every aspect, except for K and D, where *San Francisco-BART* has significantly higher values. In general we observe the same trend of degraded performance as fragmentation moves away from the found optimal point, but with varying degrees of degradation. For example in the case of *Nairobi-SACCO* where almost no degradation can be perceived.

The bottom left plot of figure 6, we have a set of 8 PT networks with total amounts of connections ranging between 51,000 and 350,000. Half of the networks show an optimal fragmentation of 1,000 connections/fragment, with the exceptions of *New Zealand-Bus* and *New York-MTABC* with 300, and *France-SNCF* and *EU-Flixbus* with 3,000. *New Zealand-Bus* shows the worst performance followed by *Amsterdam-GVB*, *New York-MTABC* and *Brussels-STIB*. They also show lower degradation when moving away from their optimal fragmentation, compared to the rest of the networks. Comparing them to the more performant *Belgium-NMBS*, *France-SNCF* and *EU-Flixbus* we can see that all are significantly larger in terms of connections. Moreover, they are all urban networks (*New Zealand-Bus* operates in Auckland and surroundings) which could mean higher number of possible walking transfers, compared to national and international networks. *London-Tube* is a particular case since it is also a urban network and has a comparable size to *New York-MTABC* and *Brussels-STIB*, but similar performance as *Belgium-NMBS*. For *London-Tube* we see again the presence of significantly high values of K and D.

Lastly, on the bottom right plot in figure 6 we see the results for the remaining 5 networks. These are the biggest networks in the set with total number of connections ranging from 689,000 to 1.2 million. In this case we see almost no degradation away from the found optimal fragmentation, with a small exception of *Madrid-CRTM*. Three of the networks, namely *Chicago-CTA*, *Helsinki-HSL* and *Flanders-De Lijn* share an optimal fragmentation of 3,000 connections/fragment. *Wallonia-TEC* was the only network of the whole set that gave its smallest possible fragment size of 1,207 connections/fragment as its optimal fragmentation.

In Figure 7 we present the (90th percentile) query respose times, measured using the optimal fragmentation found for each network (as seen on fig 6). An annotation can be seen next to every network's bar indicating the time (in ms) needed to answer 90% of the

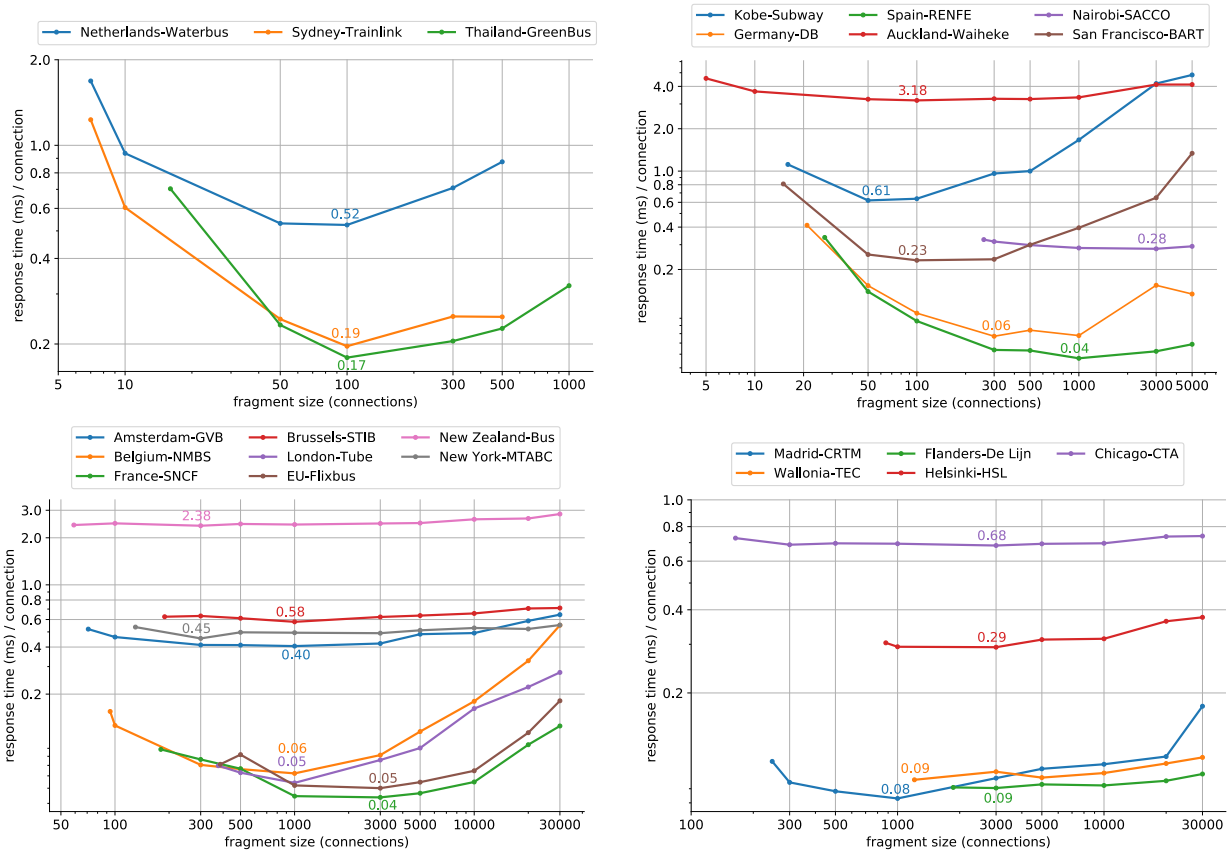

Fig. 6. Normalized response time needed to process one connection (ms/connection) vs fragment size (connections) for each PT network. PT networks with similar total number of connections (see Table 3) are grouped together to facilitate visualizing the results. We labeled the lowest point of each curve where be best performance is achieved. Axes are set with logarithmic scales.

queries of the query sets. At first glance we can see that bigger networks in terms of total number conenctions are less performant. However, *London-Tube* and *New Zealand-Bus* stand as execptions on both sides of the spectrum for this trend. *London-Tube* is a relatively big network (321,000 connections) with subsecond performance and *New Zealand-Bus* is a medium size network (153,000 connections) with much worse performance (28s) compared to networks of similar size.

5.3. Network properties and query performance

Results on how the different graph network metrics relate with route planning query performance can be seen on figure 8.

The graph that corresponds to number of stops vs query performance (upper left), shows a clear linear relation between both variables. Higher amount of stops translates into higher response times. A similar behavior can be observed for number of connections vs query performance (upper righ). More connections also means worse performance with a few outlier exceptions. *London-Tube (n9)* and to a lesser extent *Belgium-NMBS (n8)*, show better performance than other networks with similar or even less amount of connections. The opposite behavior is shown both by *Nairobi-SACCO (n10)* and *New Zealand-Bus (n18)* with significant worse performance compared with their peers.

Varied results are shown for K vs query performance (center left). Values of K show a high disperssion for both high and low performance networks, suggesting no correlation between these two variables. In the case of D vs query performance (center right), a slight proportionally inverse trend can be seen. Lower values of $D * 1000$ (< 1) seem to be a sign of higher query response time, with the exceptions of *Sydney-Trainlink (n2)* and *Auckland-Waiheke*. In contrast, higher values

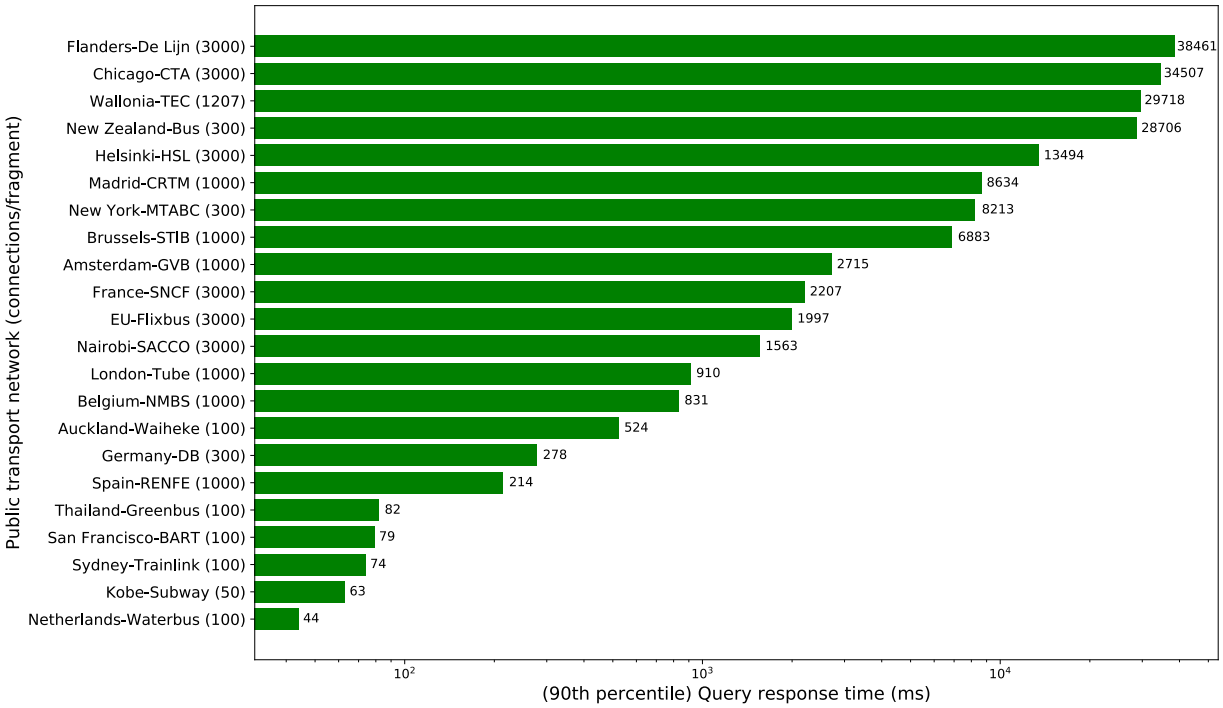

Fig. 7. Measured response time (90th percentile) in milliseconds for the fragmentation that rendered the shortest time/connection for each PT network. X-axis uses a logarithmic scale.

of $D * 1000$ (> 10) render subsecond response times, with the sole exception of *EU-Flixbus*.

Similar to K, C vs query performance (lower left) shows high disperssion of values and no indication of an existing correlation. In the case of *ACD* vs query performance (lower right), we can see that networks with the worst performance (> 10s) always show relatively low *ACD* (< 3 min). The inverse pattern can also be seen, where most networks with high *ACD* (> 10 min) show subsecond performance with the exceptions of *EU-Flixbus* and *France-SNCF* with close performance values of 1.5s and 1.9s respectively.

6. Discussion

We addessed the problem of publishing live and historical PT data in a cost-efficient way. For this, we defined a reference architecture and implementation that extends the LC approach for publishing planned schedules. Our approach handles PT schedule requests that include live updates efficiently, and is capable of providing the historical stream of events that ocurred on a PT network. This constitutes an important innovation and contribution that may be used to support for example machine learning-based applications, able to accurately predict the future state on a network.

We also studied how our approach could be used to support route planning use cases and how data fragmentation impacts the performance of query evaluation. Furthermore we measured and analyzed different topological characteristics of the 22 real PT networks used on this evaluation. We aimed on understanding what factors that drive better or worse performance.

6.1. Publishing live and historical public transport data

When it comes to live data publishing, our approach is fundamentally different to traditional data interfaces, on where data integration of planned schedules and updates take place. Traditionally, data publishers expose a stream/feed API of live data updates using SIRI or GTFS-realtime data models. Besides the request limitations these APIs normally impose to avoid server overloads, this also transfers all the computational burden of data integration and reconciliation to data reuser applications, that need to update their internall databases

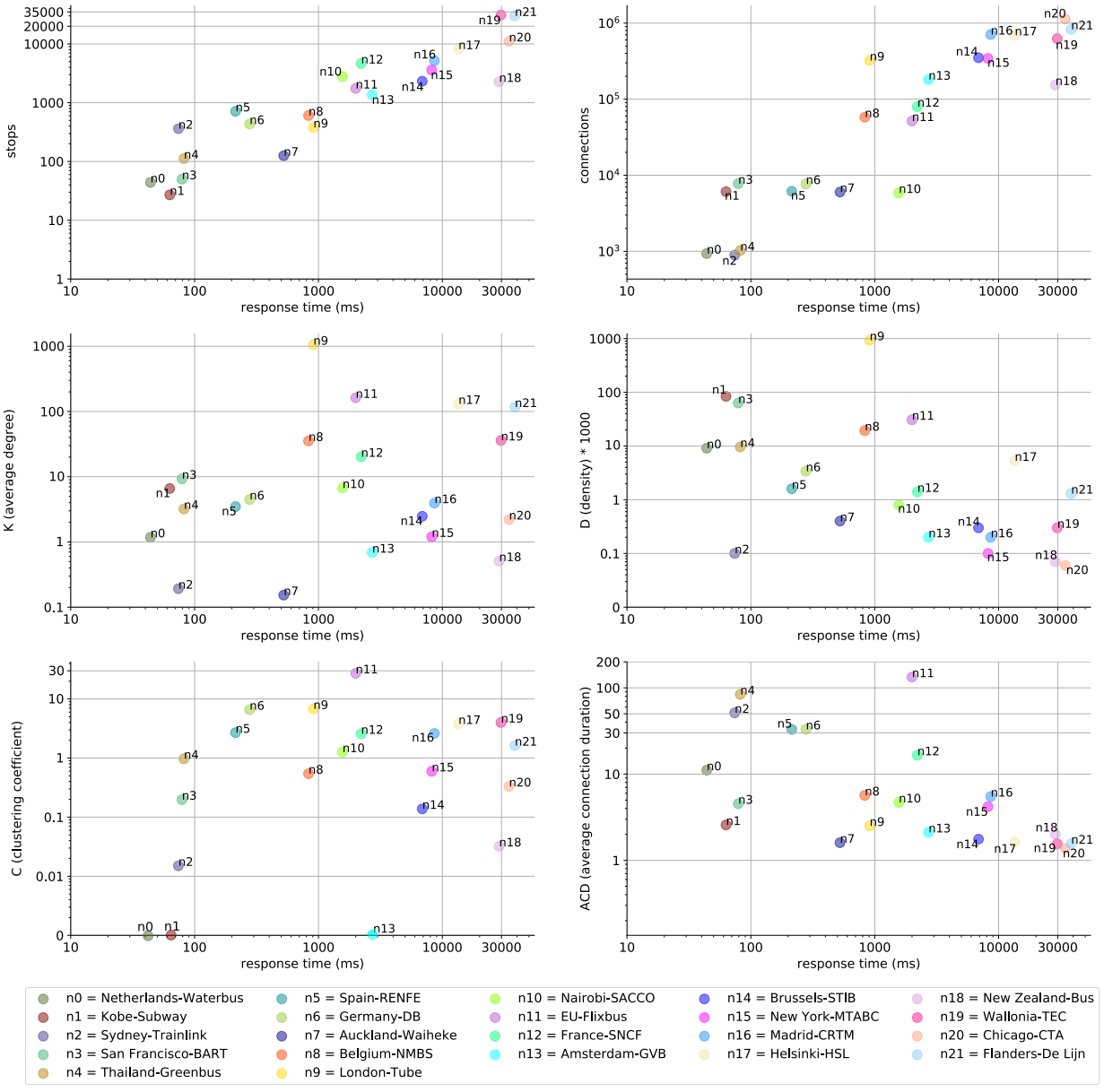

Fig. 8. PT network metrics compared with route plannning query performance. Each sub-graph compares one of the metrics to the best query evaluation performance measured for each network (see figure 7.) Axes are set in logarithmic scale.

to reflect the new schedule changes. In contrast, our approach performs such integration in an efficient way on the API server-side, by using a in-memory AVL tree data structure. This added to the planned schedule fragmentation approach of LC, allows for cost-efficient publishing of dynamic PT schedules directly from the source, while freeing data reuser applications form expensive data reconciliation tasks.

When we take a look on PT historical data we can see that is not available for most PT networks as open data on the Web. The closest data available are older versions of planned schedules for some PT networks. However, live data updates are never recorded and get lost after being breifly published through traditional APIs. Our approach enables not only to keep a historical record of previous versions of planned schedules

but also of the update stream flow as it occured. More importantly, it also defines a query interface for this data, built using standard time-based content negotiation protocol over HTTP, which provides easy access to historical data with high granularity.

6.2. Fragmentation and query evaluation performance

Our evaluation showed that for each PT network the best performance is achieved with a certain fragmentation size. This constitutes an important finding for data publishers that should be considered when designing PT data APIs. Resutls also showed that either increasing or decreasing the fragmentation size, degrades performance of route planning query evaluation. Smaller fragmentations than the optimal point, degraded performance faster than larger ones. This could be attributed to the higher number of HTTP request-response cycles needed with smaller fragments. Larger fragment sizes require clients to perform fewer cycles but need to process increased number of irrelevant connections for each query.

Regarding optimal fragmentation size, we could see that is largely related with the average number of scanned connections of the query set ($E(\text{SCQ})$). Despite the existence of a few exceptions, we saw that in general, for $E(\text{SCQ}) < 1000$ optimal fragmentation size converges on 100 connections/fragment. Similarly, for $1000 < E(\text{SCQ}) < 30000$ we saw that the optimal size converges on 1,000 connections/fragment. Finally for $E(\text{SCQ}) > 30000$ the optimal size converges on 3,000 connections/fragment. This is an important indicator to better design PT data APIs for route planning use cases. Whenever is possible to anticipate the type of queries that a PT transport network may expect, for example, by means of population distribution data or previosuly recorded query logs, fragmentation may be adjusted to render better performance overall.

Another important finding of this evaluation is the fact that for several networks, this approach results impractical for real scenarios. Response times on the order of seconds. and even tens of seconds per query are unacceptable for user applications. However is important to mention that for this evaluation we did not use any form of caching (server nor client-side), which is one of the fundamental features of publishing pattern-based fragmented datasets. Our goal was to study the impact that data fragmentation has on query performance. Different fragmentation setups influence the effort that server-side data interfaces make to respond to data fragment requests, as well as the effort made by client-side route planners for evaluating queries. We wanted to quatify these efforts show and both server and client-side caches would have hidden the effect of any fragmentation. A server-side cache frees the data interface of having to retrieve, parse and format data fragments more than once across queries. A client-side cache makes unecessary to request data fragments fetched on previous queries. Therefore, the results of this evauation may be considered as worst-case scenario values and it could be expected that in practical scenarios query response times will be significantly improved.

6.3. Network metrics and query evaluation performance

When looking at the topological properties of every PT network, we first observe the existence of a direct dependency between the size of the network, both in terms of stops and connections and the query response time. This is not a surprising result since is expected that in the presence of more connections, route planning algorithms would need to process more data for evaluating queries. Yet, it is interesting to look into individual cases that differ significantly (for better or worse) in query evaluation performace, compared with networks of similar characteristics. Examples of this atypical behavior in the case of total number of connections are *London-Tube* or *Belgium-NMBS* for better performance and *Nairobi-SACCO* or *New Zealand-Bus* for worse performance.

We see no apparent correlation for the cases of average degree K and clustering coefficient C, suggesting that these properties do not influence route planning performance. However we do observe apparent inverse relations for the cases of density D and average connection duration ACD with respect to query response times.models independent we see a pattern where for most networks with high values of ACD response times are lower compared with networks with less ACD. An explanation for this behavior could be given from the fact that lower connection duration is usually a reflection of more closely located stops, as it is for urban networks. This causes an extra processing load on the algorithm, that needs to calculate walking distances for each nearby stop and include them into the potential alternatives for route solutions. *London-Tube* stands as an outlier exception to this pattern. The reason for this is that this network groups multiple platforms as single stops, which facilitates transfer calcu-

lations for the algorithm and also serves as an explanation of the high values measured on all metrics for this network.

7. Conclusions and Future Work

Our work stands as a contribution for the PT domain as a cost-efficient data publishing alternative that includes live schedule updates and access to granular historical data. We propose a different approach on how live data is published, compared to traditional APIs, that facilitates data reuse for client appplications without sacrificing cost-efficiency on the server-side. At the same time we enable data publishers to also share historical data, which still remain largely unavailable. From a general perspective, this work provides evidence

The use of semantic Web technologies establishes a framework for data interoperability on the PT domain. Ideally, every PT operator would publish and maintain stable identifiers for their resources such as stops, routes and connections. This way, semantic interoperability starts at the source, where all derived datasets can reuse the same identifiers. This approach not only semantically models the fundamental entities and concepts of PT schedules and its updates, but also the interfaces that give access to the data. Relying on the Web infrastructure to model entire datasets as collections of HTTP resources, while including metadata that semantically describes the access patterns to traverse the collection and find more data, constitute an important contribution for the PT domain, but also sets an example that may be applied on different domains or use cases. We hope an approach such Linked Connections inspires PT organizations to publish their raw base data as a back-bone for other mobility operators in their region.

Considering the importance of query evaluation performance for supporting practical use cases, we evaluated our approach looking into understanding the factors that drive high performance and how API design can be adjusted to achieve it. With regards to our first research question **RQ1**, we conclude to accept its associated hypothesis **H1**, which constitutes an important result that highlights how careful design of API data structures can affect and improve the performance of use case-specific query evauation. Regarding research question **RQ2**, we also conclude to accept its associated hypothesis **H2**. Results showed that size in terms of stops and connections is highly correlated to rendered query evaluation performace. We could also see that density and average connection duration can also provide an indication of the expected performance. Additionally, we observed that an external factor as the number of scanned connections of expected queries could also indicate API design as the size of data fragmentation.

An important finding of this work is the confirmation that the approach performs well enough to be used in practical scenarios for most of the evaluated PT networks. However it was also evident that larger networks still remain unfeasible to be published and used in practical scenarios using this approach. Further research is needed to optimize use case driven query performance without compromising on cost-efficiency for both data publishers and reusers. On-going and future research is investigating alternative and additional fragmentation possibilities than merely time-based, such as geospatially [52]. The results shown in this paper are instrumental in that regard, as they also give an idea about the network size where high performance can still be expected and thus how larger networks could be further fragmented so that individual sub-networks render acceptable query evaluation performance.

Acknowledgements

We would like to thank David Chaves Fraga and Oscar Corcho for pushing us forward to write this paper.

References

[1] T. Davies, S.B. Walker, M. Rubenstein and F. Perini, *The State of Open Data: Histories and Horizons*, African Minds, 2019. ISBN 9781928331957.

[2] P. Colpaert and J. Rojas, Open Data Sectors and Communities: Transport, in: *The State of Open Data: Histories and Horizons*, African Minds, 2019. doi:10.5281/zenodo.2677833.

[3] B. Hogge, Transport for London Get Set, Go!, Technical Report, GovLab, 2016.

[4] G.K. Bailey, The Case for Open Data in Public Transport, UITP, 2020. https://www.uitp.org/news/the-case-for-open-data-in-public-transport/.

[5] Ably, The Maturity of Public Transport APIs 2019, Technical Report, 2019. https://files.ably.io/research/whitepapers/the-maturity-of-public-transport-apis-2019-ably-realtime.pdf.

[6] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck and P. Colpaert, Triple Pattern Fragments: a Low-cost Knowledge Graph Interface for the Web, *Journal of Web*

Semantics **37–38** (2016), 184–206. doi:doi:10.1016/j.websem.2016.03.003. http://linkeddatafragments.org/publications/jws2016.pdf.

[7] P. Colpaert, R. Verborgh and E. Mannens, Public Transit Route Planning Through Lightweight Linked Data Interfaces, in: *Web Engineering*, J. Cabot, R. De Virgilio and R. Torlone, eds, Springer International Publishing, Cham, 2017, pp. 403–411. ISBN 978-3-319-60131-1.

[8] P. Colpaert, A. Llaves, R. Verborgh, O. Corcho, E. Mannens and R. Van de Walle, Intermodal public transit routing using Linked Connections, in: *Proceedings of the 14th International Semantic Web Conference: Posters and Demos*, CEUR Workshop Proceedings, Vol. 1486, 2015. ISSN 1613-0073. http://ceur-ws.org/Vol-1486/paper_28.pdf.

[9] C. Bizer, T. Heath and T. Berners-Lee, Linked data-the story so far, *International journal on semantic web and information systems* **5**(3) (2009), 1–22.

[10] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web, *Scientific American* **284**(5) (2001), 34–43. http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[11] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space (2011). http://linkeddatabook.com/editions/1.0/.

[12] R. Verborgh, M. Vander Sande, P. Colpaert, S. Coppens, E. Mannens and R. Van de Walle, Web-Scale Querying through Linked Data Fragments, in: *Proceedings of the 7th Workshop on Linked Data on the Web*, C. Bizer, T. Heath, S. Auer and T. Berners-Lee, eds, CEUR Workshop Proceedings, Vol. 1184, 2014. ISSN 1613-0073. http://ceur-ws.org/Vol-1184/ldow2014_paper_04.pdf.

[13] Y. Li and T. Voege, Mobility as a Service (MaaS): Challenges of Implementation and Policy Required, *Journal of Transportation Technologies* **7** (2017), 95–106. doi:10.4236/jtts.2017.72007.

[14] D. Chaves-Fraga, A. Antón, J. Toledo and Ó. Corcho, ONETT: Systematic Knowledge Graph Generation for National Access Points, in: *SEM4TRA-AMAR@SEMANTICS*, 2019.

[15] M. Katsumi and M. Fox, Ontologies for transportation research: A survey, *Transportation Research Part C: Emerging Technologies* **89** (2018), 53–82. doi:https://doi.org/10.1016/j.trc.2018.01.023. http://www.sciencedirect.com/science/article/pii/S0968090X18300858.

[16] S.K. Fayyaz S., X.C. Liu and G. Zhang, An efficient General Transit Feed Specification (GTFS) enabled algorithm for dynamic transit accessibility analysis, *PLOS ONE* **12**(10) (2017), 1–22. doi:10.1371/journal.pone.0185333.

[17] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez and D.-U. Hwang, Complex networks: Structure and dynamics, *Physics Reports* **424**(4) (2006), 175–308. doi:https://doi.org/10.1016/j.physrep.2005.10.009. http://www.sciencedirect.com/science/article/pii/S037015730500462X.

[18] T. Tsekeris and A.-Z. Souliotou, Graph-theoretic evaluation support tool for fixed-route transport development in metropolitan areas, *Transport Policy* **32** (2014), 88–95. doi:https://doi.org/10.1016/j.tranpol.2014.01.005. http://www.sciencedirect.com/science/article/pii/S0967070X14000146.

[19] D. Gattuso and E. Miriello, Compared Analysis of Metro Networks Supported by Graph Theory, *Networks and Spatial Economics* **5** (2005), 395–414.

[20] H. Soh, S. Lim, T. Zhang, X. Fu, G.K.K. Lee, T.G.G. Hung, P. Di, S. Prakasam and L. Wong, Weighted complex network analysis of travel routes on the Singapore public transportation system, *Physica A: Statistical Mechanics and its Applications* **389**(24) (2010), 5852–5863. doi:https://doi.org/10.1016/j.physa.2010.08.015. http://www.sciencedirect.com/science/article/pii/S0378437110006977.

[21] C. Chen, T. D'Alfonso, H. Guo and C. Jiang, Graph theoretical analysis of the Chinese high-speed rail network over time, *Research in Transportation Economics* **72** (2018), 3–14, Long-distance passenger transport market. doi:https://doi.org/10.1016/j.retrec.2018.07.030. http://www.sciencedirect.com/science/article/pii/S0739885917303141.

[22] P. Fortin, C. Morency and M. Trépanier, Innovative GTFS Data Application for Transit Network Analysis Using a Graph-Oriented Method, *The Journal of Public Transportation* **19** (2016), 2.

[23] A. Galati, V. Vukadinovic, M. Olivares and S. Mangold, Analyzing temporal metrics of public transportation for designing scalable delay-tolerant networks, in: *PM2HW2N '13*, 2013.

[24] M. Kurant and P. Thiran, Extraction and analysis of traffic and topologies of transportation networks, *Phys. Rev. E* **74** (2006), 036114. doi:10.1103/PhysRevE.74.036114.

[25] P. Sen, S. Dasgupta, A. Chatterjee, P. Sreeram, G. Mukherjee and S. Manna, Small-world properties of the Indian railway network., *Physical review. E, Statistical, nonlinear, and soft matter physics* **67 3 Pt 2** (2003).

[26] S. Derrible and C. Kennedy, Network Analysis of World Subway Systems Using Updated Graph Theory, *Transportation Research Record* **2112**(1) (2009), 17–25. doi:10.3141/2112-03.

[27] C. von Ferber, T. Holovatch, Y. Holovatch and V. Palchykov, Public transport networks: empirical analysis and modeling, *The European Physical Journal B* **68** (2009), 261–275.

[28] T. Shanmukhappa, I.W.-H. Ho and C.K. Tse, Spatial analysis of bus transport networks using network theory, *Physica A: Statistical Mechanics and its Applications* **502** (2018), 295–314. doi:https://doi.org/10.1016/j.physa.2018.02.111. http://www.sciencedirect.com/science/article/pii/S0378437118302024.

[29] A. Erath, M. Löchl and K. Axhausen, Graph-Theoretical Analysis of the Swiss Road and Railway Networks Over Time, *Networks and Spatial Economics* **9** (2009), 379–400. doi:10.1007/s11067-008-9074-7.

[30] S. Derrible and C. Kennedy, Applications of Graph Theory and Network Science to Transit Network Design, *Transport Reviews* **31**(4) (2011), 495–519. doi:10.1080/01441647.2010.543709.

[31] H. Lu and Y. Shi, Complexity of Public Transport Networks, *Tsinghua Science & Technology* **12**(2) (2007), 204–213. doi:10.1016/S1007-0214(07)70029-9. http://www.sciencedirect.com/science/article/pii/S1007021407700299.

[32] J. Hong, R. Tamakloe, S. Lee and D. Park, Exploring the Topological Characteristics of Complex Public Transportation Networks: Focus on Variations in Both Single and Integrated Systems in the Seoul Metropolitan Area, *Sustainability* **11** (2019), 5404. doi:10.3390/su11195404.

[33] C. Guéret, P. Groth, C. Stadler and J. Lehmann, Assessing Linked Data Mappings Using Network Measures, in: *The Semantic Web: Research and Applications*, E. Simperl, P. Cimi-

ano, A. Polleres, O. Corcho and V. Presutti, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 87–102.

[34] H. Bast, D. Delling, A.V. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner and R.F. Werneck, Route Planning in Transportation Networks, *CoRR* **abs/1504.05140** (2015). http://arxiv.org/abs/1504.05140.

[35] T. Pajor, Algorithm Engineering for Realistic Journey Planning in Transportation Networks, PhD thesis, 2013. https://d-nb.info/1058165240/34.

[36] E.W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematik* (1959), 269–271. ISBN 0945-3245. doi:10.1007/BF01386390.

[37] G. Stølting Brodal and R. Jacob, Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries, *Electronic Notes in Theoretical Computer Science* **92** (2004), 3–15, Proceedings of ATMOS Workshop 2003. doi:https://doi.org/10.1016/j.entcs.2003.12.019. http://www.sciencedirect.com/science/article/pii/S1571066104000040.

[38] E. Pyrga, F. Schulz, D. Wagner and C. Zaroliagis, Efficient Models for Timetable Information in Public Transportation Systems, *ACM J. Exp. Algorithmics* **12** (2008). doi:10.1145/1227161.1227166.

[39] D. Delling, J. Dibbelt and T. Pajor, Fast and Exact Public Transit Routing with Restricted Pareto Sets, in: *Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments, ALENEX 2019, San Diego, CA, USA, January 7-8, 2019.*, 2019, pp. 54–65. doi:10.1137/1.9781611975499.5.

[40] J. Dibbelt, T. Pajor, B. Strasser and D. Wagner, Connection Scan Algorithm, *J. Exp. Algorithmics* **23** (2018), 1.7:1–1.7:56. doi:10.1145/3274661.

[41] H. Bast, M. Hertel and S. Storandt, Scalable Transfer Patterns, in: *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2016.

[42] S. Witt, Trip-Based Public Transit Routing Using Condensed Search Trees, in: *ATMOS*, 2016. https://arxiv.org/pdf/1607.01299.pdf.

[43] D. Delling, T. Pajor and R.F. Werneck, Round-Based Public Transit Routing, in: *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, ALENEX '12, Society for Industrial and Applied Mathematics, 2012, pp. 130140–.

[44] L. Heppe and T. Liebig, Real-Time Public Transport Delay Prediction for Situation-Aware Routing, in: *KI 2017: Advances in Artificial Intelligence*, G. Kern-Isberner, J. Fürnkranz and M. Thimm, eds, Springer International Publishing, Cham, 2017, pp. 128–141. ISBN 978-3-319-67190-1.

[45] M. Berkow, A.M. El-Geneidy, R.L. Bertini and D. Crout, Beyond Generating Transit Performance Measures: Visualizations and Statistical Analysis with Historical Data, *Transportation Research Record* **2111**(1) (2009), 158–168. doi:10.3141/2111-18.

[46] H. Sompel, M. Nelson, R. Sanderson, L. Balakireva, S. Ainsworth and H. Shankar, Memento: Time Travel for the Web (2009).

[47] R. Taelman, M. Vander Sande, J. Van Herwegen, E. Mannens and R. Verborgh, Triple storage for random-access versioned querying of RDF archives, *Journal of Web Semantics* **54** (2019), 4–28, Managing the Evolution and Preservation of the Data Web. doi:https://doi.org/10.1016/j.websem.2018.08.001. http://www.sciencedirect.com/science/article/pii/S1570826818300404.

[48] J. Rojas Meléndez, D. Chaves-Fraga, P. Colpaert, R. Verborgh and E. Mannens, Providing Reliable Access to Real-Time and Historic Public Transport Data Using Linked Connections, in: *Proceedings of the 16th International Semantic Web Conference: Posters and Demos*, 2017. https://iswc2017.semanticweb.org/wp-content/uploads/papers/PostersDemos/paper637.pdf.

[49] J.A. Rojas, D. Van Assche, H. Delva, P. Colpaert and R. Verborgh, Efficient Live Public Transport Data Sharing for Route Planning on the Web, in: *Web Engineering*, M. Bielikova, T. Mikkonen and C. Pautasso, eds, Springer International Publishing, Cham, 2020, pp. 321–336. ISBN 978-3-030-50578-3.

[50] G.M. Skii and Y.M. Landis, An algorithm for the organization of information, 1962.

[51] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo and V. Latora, *Graph Metrics for Temporal Networks*, in: *Temporal Networks*, P. Holme and J. Saramäki, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 15–40. ISBN 978-3-642-36461-7. doi:10.1007/978-3-642-36461-7_2.

[52] H. Delva, J. Rojas Meléndez, P.-J. Vandenberghe, P. Colpaert and R. Verborgh, Geospatial Partitioning of Open Transit Data, in: *Proceedings of the 20th International Conference on Web Engineering*, M. Bielikova, T. Mikkonen and C. Pautasso, eds, Lecture Notes in Computer Science, Vol. 12128, Springer, 2020, pp. 305–320. ISBN 978-3-030-50578-3. doi:10.1007/978-3-030-50578-3_21.

[53] D. Brickley, M. Burgess and N. Noy, Google Dataset Search: Building a Search Engine for Datasets in an Open Web Ecosystem, in: *The World Wide Web Conference*, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 13651375–. ISBN 9781450366748. doi:10.1145/3308558.3313685.