Maduako *et al. J Big Data* (2019) 6:55
https://doi.org/10.1186/s40537-019-0218-z

Journal of Big Data

RESEARCH Open Access

STVG: an evolutionary graph framework for analyzing fast-evolving networks

Ikechukwu Maduako[1*], Monica Wachowicz[1] and Trevor Hanson[2]

*Correspondence:
iykemadu84@gmail.com
[1] People in Motion Lab, University of New Brunswick, 15 Dineen Drive, Fredericton, NB E3B 5A3, Canada
Full list of author information is available at the end of the article

Abstract

Sequence of graph snapshots have been commonly utilized in literature to represent changes in a dynamic graph. This approach may be suitable for small-size and slowly evolving graphs; however, it is associated with high storage overhead in massive and fast-evolving graphs because of replication of the entire graph from one snapshot to another at shorter temporal resolutions. This presents a drawback especially where efficient evolutionary analytics relies on the explanatory power of representing the dynamics of the graph across different temporal resolutions. In this paper, we propose a framework based on our Space–Time-varying graph (STVG) formalism which utilizes the Whole-graph approach to model the dynamics of a graph such that the evolution of the graph materializes in the time-varying changes of its Projected graphs. The STVG framework provides an approach to reduce high storage overhead in massively changing graph where new nodes and edges arrive every second. It affords the capability to extract Projected graphs at different time-windows and analyze their metrics across varying temporal resolutions. We demonstrate how the proposed STVG framework can be exploited to identify and extract evolutionary patterns in public bus transit graph using metrics such as graph density, volume and average path length. The results reveal evolutionary patterns in the overall network density, traffic congestion density as well as graph density with respect to bus movement at hourly, daily and monthly temporal resolutions.

Keywords: Space–Time varying graph, Evolutionary graph analytics, Fast-evolving networks, Transit networks

Introduction

Evolutionary graph analytics have attracted attention from many research communities with the main purpose of understanding the changing pattern of real-world networks through evolutionary analysis of graph metrics and dynamic interactions between entities. Graphs of real-world networks evolve as new nodes and edges continually appear and disappear in the structure but, more importantly, their metrics such as density, average path length and network diameter also evolve. Uncovering and understanding hidden patterns in an evolving network requires evolutionary analysis of the network over different temporal resolutions. Evolutionary graph analytics have been explored for use in different types of networks including web citation and co-authorship networks [1–4], online social networks [5–10], biology and disease networks [11–14], as well as in communication networks [15–20]. All networks do not evolve at the same rate; some

evolve swiftly while others evolve slowly [21]. For example, in transit and mobility networks, nodes and edges are added to their graph on the temporal resolution of seconds, whereas in co-authorship networks, nodes and edges are added to their graph on the temporal resolution of months or years. Correspondingly, these cases require different approaches for the evolutionary analysis.

Networks whose graphs evolve slowly over time and space can be effectively analyzed through a sequence of graph snapshots where snapshots of the graph at two distinct timestamps t_1, and t_2, can be the basis for change extraction. The entire graph can be replicated from one snapshot to another with few computational and storage limitations because of the coarse temporal resolution. The challenge is that for fast-evolving networks, where the evolutionary analysis relies on the explanatory power of representing the changes in the network across smaller temporal resolutions, the snapshot method can become ineffective due to high computation and storage demands. An approach that reduces replication of the entire graph from one time-window to another to avoid high storage overhead even as the graphs grow at different temporal resolutions. The Whole-graph approach presents a method where the entire evolution of a graph through time is captured in an unabridged, but complete, graph that evolves compactly as nodes and edges appear and disappear in the graph. Each node or edge has a timestamp depicting its valid time in the graph. The valid time captures when the insertion of new nodes, edges and attributes or deletion of nodes, edges and their attributes are made in the graph.

Longer processing time involved in extracting time-dependent subgraphs from the Whole-graph can be a disadvantage unlike the convenience that snapshot method provides through versioning of smaller graph volume at each time-window. This occurs majorly when the time dimension of a Whole-graph is represented by timestamps that are stored as attributes of the nodes or edges. Updates and query processing take longer processing time because there is a massive number of node and edge attributes to be scanned. This bottleneck can be reduced by representing the time dimension as a graph such as using a time-graph [22]. The advantages of the Whole-graph approach over the snapshot method include no replication of static nodes and edges across time which leads to less storage overhead. Also, longitudinal queries across time are less complex than in the snapshot approach because a simple time-dependent query can run through the entire graph, retrieve and compare the state of the graph across different temporal resolutions.

In this paper, we present the space-time varying graph (STVG) framework for evolutionary graph analytics of fast-evolving real-world networks. The underlying graph G in our STVG is modeled based on the Whole-graph approach where the network structure evolves in space and time in such a way that evolutionary patterns are due to the changes in the connectivity and adjacency relationships among nodes in the network. The graph grows continuously as new nodes and edges emerge through time. The Whole-graph is composed of subgraphs including the time-graph. Subgraphs are used to facilitate the conceptual modelling of connectivity between entities in distinct spaces of the real-word network. The subgraphs are connected to the time-graph to keep track of the evolution of the Whole-graph. Projected graphs retrieve the state of the Whole-graph at any given temporal resolution. The evolution of the Whole-graph becomes visible in the

space–time varying changes of the Projected Graphs. Relevant graph metrics are used to retrieve Whole-graph evolution from its Projected Graphs. The STVG therefore, is actualized from the Whole-graph, Subgraphs and Projected graph formalism.

We applied the STVG framework for evolutionary graph analytics of dynamic transit network to uncover the evolving behavior of the network at various temporal resolutions. The dynamic interactions between places and mobility characteristics of moving vehicles in a transit network presents an evolving graph that changes swiftly in structure and size [23]. The analytical results show the potential of using the framework for transit and mobility trend analysis with possible application to time-dependent transit recommendation systems.

Our research challenge consists of processing a vast volume of transit feeds continuously coming at high velocity from a large fleet of buses, but also making sure that our STVG framework captures the evolving behavior of subgraphs belonging to a Whole-graph at different time resolutions. Towards this challenge, our scientific contributions are as follows:

- Previous research on evolving graphs has focussed on small complex networks. This paper contributes to the emerging research work on massive and fast-evolving networks.
- Real-world networks such as transit networks change at shorter temporal and spatial resolutions, and the proposed STVG framework explores how to capture and analyse the evolution of fast-evolving graphs at different time resolutions.
- This paper also contributes to the use of graph databases and graph queries to store and capture the evolutionary behavior of fast-evolving networks at different spatial and temporal resolutions with less storage overhead and complex query processing.

The remaining of this paper is organized as follows: "Related work" section summarizes the previous research work on graph data analytics. "Modelling methods" describes the modeling process of our approach, including the overall framework and the details. "Experimental study" section presents the experimental study and the discussion of results. Finally, conclusions are presented in "Conclusion" section.

Related work

Previous research work has applied the snapshot approach to real-world networks such as the social, web-citation and co-authorship networks. In [24], a survey was carried out on time-evolving graphs of a social network with emphasis on the temporal metrics. They propose a general formalism to study the evolution of the temporal metrics based on a sequence of snapshots across time-windows. Graph temporal metrics such as density, average path, proximity, reachability time, centralities, clustering coefficient and conductance were discussed in this study.

In contrast, Huo and Tsotras [25] focused on the problem of efficient temporal shortest path queries on graph snapshots of a social network. The study utilized fixed time-window partitioning on a slowly evolving graph to generate a sequence of snapshots on which the queries are applied, one at a time. They had a total of 165 daily graph snapshots of a social network graph and adjusted the query time-windows to 5-day and

25-day intervals. The concept of temporal shortest path in this study is noted to be useful for user's historical trend analysis, for example, to discover how close two users were in the past and how their closeness has evolved over time. This work utilizes ad-hoc query over a sequence of graph snapshots, but the major drawback is high storage overhead.

Some strategies have been proposed to address the storage overhead issue by using a set of deltas that store only the changes that are needed to construct a snapshot. The most efficient compression approach is known as SM-FVF which creates clusters of k snapshots, where a compressed graph has no redundancy among delta files of the same cluster, but it still has redundancy between delta files of different clusters [26]. Meanwhile, Gottumukkala et al. [24] proposes the possibility of integrating Neo4J and Spark's GraphX in a workflow to optimize the storage, processing and visualization of big graphs. They utilized this method to carry out visual analytics of evolving social network in graph snapshots to track the evolutionary state of the graph including the structure and topology.

Lerman et al. [27] worked on a different formulation of centrality metric for evolving graph analysis that measures the number of paths that exist between source and destination nodes over time in a graph. Their model was based on sequence of graph snapshots and it was evaluated on a scientific papers' citation graphs whose dynamics occurs slowly. It was concluded in this paper that centrality measures of some articles were under or over estimated by previous studies that did not take into account the evolutionary nature of the graph. The evolution of essential genes (central nodes) and their roles in a cell survival and development were studied in the work of Jalili et al. [14], utilizing node centrality evolution in a sequence of graph snapshots of a biological network. Quattociocchi et al. [28] utilized the time-varying graph framework proposed by Casteigts et al. [23] based on a sequence of graph snapshots to study the dynamics in coexistence of co-authorship and citation network. The graph snapshots reveal the evolution in a scientific network extracted from a portion of the arXiv repository covering a period of 10 years of publications in Physics. It shows that scientific networks are dynamic as nodes (the scientist) join, participate, attract, compete, cooperate and disappear in the network which affects the shape and strength of the graph's connectivity. They discovered how the selection process of citations may affect the shape of the co-authorship network from a sparser and disconnected structure to a dense and homogenous one.

While the sequence of graph snapshot method is associated with disadvantages of high storage overhead and computational complexity, the Whole-graph approach provides an alternative. The Whole-graph approach represents graph evolution not in sequence of static snapshots but as an unabridged graph that evolves compactly as nodes and edges appear and disappear in the graph. This approach was utilized in Pereira et al. [8] to analyze evolving centrality metrics in a dynamic twitter graph. The study reveals how closeness and betweenness centrality measures evolve in the follower/followed relationship. They found that the twitter network is dynamic, users can assume or leave central roles in the graph through time. However, the time-dimension of the graph is based on timestamp attributes of the nodes and edges of the graph. Each node and edge in the graph have a valid time at which they existed in the graph. Varying-time windows of weekly, fortnightly and monthly windows were used in the analysis of the graph's evolution. As

earlier mentioned, having timestamps as attributes of the nodes and edges is inefficient in cases where there are billions of nodes and edges to scan through. Meanwhile, Kumar et al. [29] similarly examined the evolution of Flickr and Yahoo 360 social graph in parallel. They found out similarities in the evolution pattern and properties of the two different graphs. The whole-graph approach was also utilized in burst area discovery based on evolving top-k changes in a stream of fast bipartite graph evolutions of users and stories on the Digg.com website [30]. The authors, however, utilized the Haar wavelet tree to reduce high computation complexity associated with a fast-evolving graph stream. Top-k burst areas are computed incrementally from small hop size to large hop size of time windows.

Recently, Qiangjuan et al. [1] proposed the supra-adjacency matrix approach to define a temporal network used to analyze the evolution of important nodes in Enron email communication network and DBLP co-authorship network. They treated temporal networks as a special case of multi-layer network where the emerging graph G is defined as a graph of N nodes divided into T time-windows which therefore, creates a supra-adjacency matrix of NT x NT dimensions. This approach could be suitable for a slow evolving graph whose properties changes at longer time intervals, such as in co-authorship networks but will face massive computational changes in a mobility graph that evolves, for example every 5 s. Supra-adjacency matrix created at seconds, minutes and even hourly time-windows for a period of 18 months or more will face huge computational overhead.

Overall, these studies represent a growing interest in evolutionary analytics of dynamic networks. However, the focus has been on slow evolving social graphs, dealing majorly with small snapshots of graphs. Real-world networks such the transit network and communication network however present fast-evolving graphs that change at shorter temporal and spatial resolutions. We therefore, present an evolutionary graph analytics framework that is based on the Whole-graph approach to capture evolutionary patterns at various temporal resolutions. The STVG framework provides an approach to reduce high storage overhead and affords us the ability to extract Projected graphs at different time-windows and analyze their metrics across varying temporal resolutions.

Modelling methods

This section describes the main components of the STVG framework and the methodology for modeling and capturing the evolution of the graphs.

The main components of our STVG framework

In this section, we present our STVG framework which consists of three main components as illustrated in Fig. 1:

- *Whole-graph* represents the entire content of a STVG in a given lifetime. The main purpose is to store all data based on an unabridged graph that grows and evolves through time rather than graph snapshots.
- *Subgraphs* represent entities and their relationships in distinct non-metric spaces (e.g. place, event, mobility, and social spaces) of the Whole-graph. The main purpose

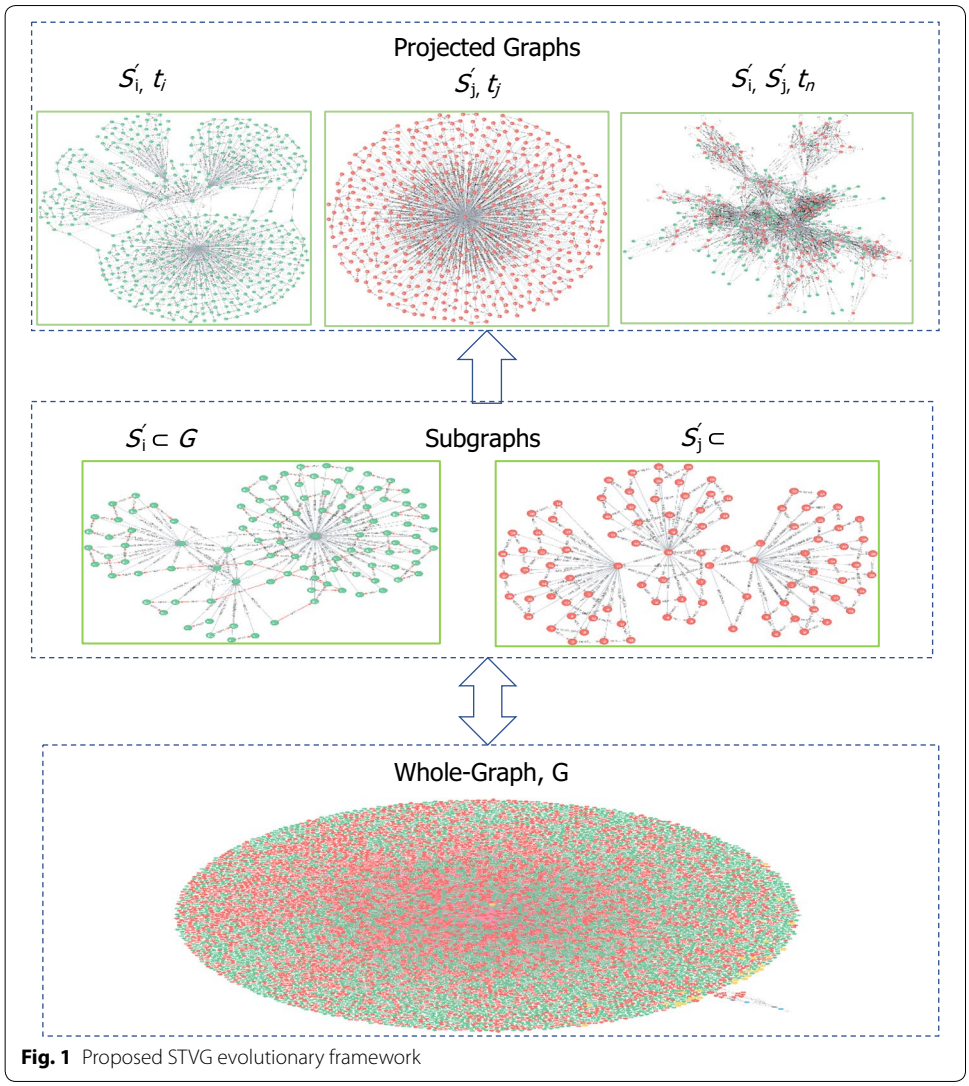


Fig. 1 Proposed STVG evolutionary framework

is to facilitate the conceptual modelling of the connectivity between entities in distinct spaces that generate the Whole-graph.

– *Projected graphs* represent nodes and edges that are retrieved at varying time-windows and temporal resolution as well as based on subgraph of interest that will be used in the evolutionary analytics of the Whole-graph.

Figure 1 illustrates the main components of our STVG framework: Whole-graph, Subgraphs and Projected graphs. The whole-graph, G consists of all nodes and edges belonging to multiple subgraphs, $\left(S'_i, S'_j, \ldots S'_n\right)$. In this case, S'_i represents the nodes and edges of a mobility space, meanwhile S'_j represents the nodes and edges of a geographical space. The Projected graphs at any timestamp can consist of one subgraph of interest as illustrated in timestamps t_i and t_j, as well as multiple subgraphs at the same timestamp such as shown in t_n in Fig. 1.

There are conceptual challenges in representing the dynamic behavior of a real-world network, especially where the evolutionary analysis relies on the explanatory power of

representing interactions in the network across different spatial and temporal resolutions. An efficient modelling and analytical approach should be such that reduces replication of the entire graph from one time-window to another to reduce high storage overhead. Our approach is such that combines the Whole-graph, subgraphs and Projected Graphs in a framework to model and analyze evolutions of fast-evolving networks. Essential evolutionary assumptions of the Whole-graph, Subgraphs and the Projected graphs must be put into account to uncover the dynamic properties of the STVG using classical graph metrics. We therefore, present in "The main components of our STVG framework", "Whole-graph" and "Subgraphs" sections the modeling concepts of the three components of the STVG framework and their corresponding evolutionary assumption as formulated in [23].

Whole-graph

The Whole-Graph G represents the entire content in a given lifetime $T(t_1 - t_n)$ of the STVG, whose nodes and edges grow through time. G consists of all the nodes and edges through time, $t_1 - t_n$, $[N_{[t_1-t_n]}, E_{[t_1-t_n]}]$, where $N_{[t1-tn]}$ and $E_{[t_1-t_n]}$ are sets of all node and edge instances of the Whole-graph stored in a database. Conceptually, the Whole-graph G has two dimensions namely, the Space dimension and the Time dimension which are represented as (*S, T*) where *S* is a digraph and *T* is a tree graph. *S* is composed of subgraphs $\left(S'_1,\ S'_2, \ldots S'_n\right)$, and each subgraph is a subset of the whole-graph *G* in the STVG. Each node or edge in *S* has a timestamp depicting its valid time in the graph. This is used to capture the insertion of new nodes, edges and attributes or deletion of nodes, edges and their attributes and to track the evolution of the graph G.

The evolutionary assumptions underlying the Whole-graph G are described based on the following:

- *Vertex-centric evolution* The evolution of the Whole-graph materializes in the dynamic changes within a node's neighborhood. This is an important assumption to enable the capability to extract both global and local evolution of the graph's properties.
- *Complete graph connectivity* At any point in time, a node in G must have an edge connection with an existing node. This assumption is essential to support global reachability across the entire graph which is important in querying the global evolution of a real-world network at any point in time. This is one of the advantages of capturing evolution in a dynamic graph based on the Whole-graph formalism instead of as snapshots.
- *Periodicity of nodes* The assumption holds in practice in many real-world networks. More specifically, it provides the ability to exploit STVG properties using nodal graph metrics that are generated by the dynamics of a network (e.g. recurrent existence of nodes). This assumption is also important in defining temporal resolutions at which the dynamics of graph can be discovered.

Subgraphs

Subgraphs $\left(S'_1,\ S'_2, \ldots S'_n\right)$ are used to facilitate conceptual modelling of connectivity within and between entities in distinct spaces of a real-world network. However, in the database, the graph G is not stored in subgraphs but as a Whole-graph with complete connectivity between entities. It is also important to point out that subgraphs

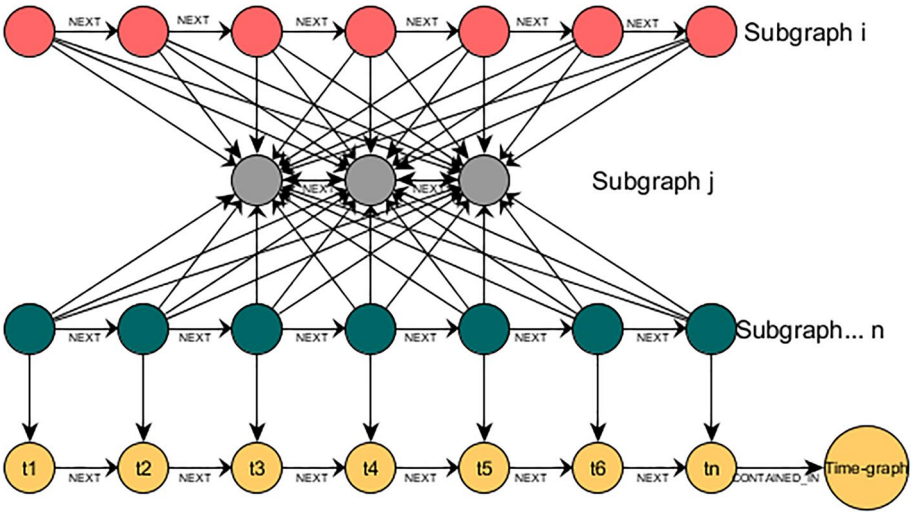


Fig. 2 Connectivity between the subgraphs and the time-graph

are not graph snapshots since they require constant and complementary connectivity (at any point in time, the nodes belonging to one subgraph are linked to the nodes belonging to another subgraph) as well as eventual adjacency (at any point in time, a target node will be sequentially linked to a source node in the same subgraph).

Multiple subgraphs S' can be defined such that $S = (S'_1, S'_2, \ldots S'_n)$ consisting of nodes and edges that belong to a specific type entity in a real-world network such as places, people, events and so on. Nodes belonging to different subgraphs $S'_i \subset S$ and $S'_j \subset S$ are linked by complementary connectivity edges since for every node in a subgraph there is always a corresponding node in another subgraph it is related to. However, only one subgraph S'_i is connected to T which is another subgraph of G (time-graph) that represents the natural levels of temporal resolutions. Nodes belonging to the same subgraph $S'_i \subset S$ are linked by eventual adjacency edges ("NEXT") between an existing node (source node) and a future node (target node) as described in Fig. 2.

Essentially, T is a hierarchical (tree) structure where a root node is followed by year nodes, month nodes, day nodes, and further temporal resolutions. Such that each node of a subgraph S'_i links to a leaf-node of T. This is an important constraint to support the capturing of vertex-centric evolution of the graph G at any defined temporal resolution. Similarly, every node in the subgraphs has a label and properties which are essential for label-based projections. Labels assign roles (e.g. "Moving" representing the state of a moving entity) and objects (e.g. "Street" and "Intersection"). Multiple properties can also be attached to the nodes and edges of a subgraph S'_i which are attributes $(a_1 \ldots a_n)$ containing semantic information about the entity.

The evolutionary assumptions underlying the Subgraphs S'_i are described based on the following:

– *Node connectivity over time* At any point in time, at least one node belonging to a subgraph S'_i can be reached by all the other nodes in S. This assumption is essential to

support any graph query/algorithm whose input is spread over all nodes generated at different time instances, with at least one node capable of generating the output.

- *No recurrence of edges over time* An edge is created only once at any point in time. This assumption is critical to maintain the eventual adjacency relationships thus to perform any computation of a graph metric such as shortest path and centrality.
- *Constant connectivity over time* Subgraphs are in constant connectivity between themselves, however, only one subgraph S_i' and T are always linked at any point in time. There must exist an edge between a node belonging to a subgraph $S_i' \subset S$ and a leaf-node of T.
- *Eventual adjacency* There is always a future target node in subgraph $S_i' \subset S$ that a source node will be linked to.

Conceptually, the subgraphs are essential for convenient definition nodes and edges that are being projected for evolutionary analysis. The Projected Graphs can be locally extracted from a single subgraph or globally from all the subgraphs.

Projected graphs

Projected graphs are vital for retrieving nodes and edges that define the state of the Whole-graph at varying temporal resolutions, time-windows and based on subgraphs of interest. Projected graphs present the key graphs on which graph metrics are computed for evolutionary analytics of the Whole-graph G. From the vertex-centric standpoint, the evolution of the Whole-graph G materializes in the time-varying changes of its Projected graphs, $PG_{(\mathcal{T})}$. Depending on the network and application, varying resolutions of time can be defined to appropriately capture the evolutionary behavior of the graph. At the extreme, each interval could correspond to the smallest time unit in the dataset or to the time between any two consecutive modifications of the graph [31]. In some cases, every Projected graph of the whole-graph G becomes equivalent to an evolving graph model [32].

Time, $\mathcal{T}$ of the Projected graph can represent temporal resolutions of the graph such as in seconds, minutes, hours, days, weeks, months and years. It can also represent time-windows such as a 2-week window, one-month window, and 2-year window. Time-windows can be defined as event-based such as to capture changes in the network through the period of a snow storm or other event.

The time-window can be a sliding window or a tumbling window depending on the application. A sliding window defines a time period that goes back in time from the present. For instance, a sliding window of 1 h includes nodes and edges that have appeared in the last 1 h. In a tumbling window, nodes and edges are grouped in a single window based on time. For example, in a tumbling window with a length of 2 h, the first window (w_1) contains nodes and edges that appeared between the zeroth and second hours. The second window (w_2) contains nodes and edges that appeared between the second and fourth hours and the third window (w_3) contains nodes and edges that appeared between the fourth and sixth hours. In this case, the Whole-graph G is projected every 2 h, and none of the windows overlap; each Projected graph represents a distinct and unique state of the Whole-graph.

The evolutionary assumptions underlying the Projected graphs can be described based on the following:

- *Recurrence of nodes* a node of a Projected Graph $PG_{(\tau)}$ can reoccur in different Projected Graphs at different time windows, $w_i, w_j \ldots w_k$, even in the case of tumbling time-windows. This assumption is important because some entities (nodes) in a real-world network (e.g. places) do not change but other entities and connectivity around them do change. The nodes will appear in more than one time-window because its neighborhood changes across time.
- *No recurrence of edges* in tumbling time-windows, an edge in a Projected graph $PG_{(t_i)}$ cannot reoccur in another Projected Graph because connectivity in each Projected graph is distinct and unique.

Capturing the evolution of STVG

Graph metrics are generally used to uncover the dynamics of the Whole-graph globally or dynamics in the graph locally over the Projected graphs. The process of capturing the STVG evolution involves a pipeline of steps that can be described as follows:

- Step 1: Define a time-window, $\Upsilon \subset \mathrm{T}$ for analysis, where Υ is the selected time-window and T is the entire lifetime of the Whole-graph.
- Step 2: Define subgraph(s) of interest, $\mathrm{S}' \subset \mathrm{S}$ where, S' is the selected subgraph (s) and S is the entire space-dimension of the whole-graph.
- Step 3: Define the temporal resolution $t_i \subset \Upsilon$ for iterative projections and for each t_i in Υ.
- Step 4: Project Graph, $PG_{(t_i)}$ where $PG_{(t_i)}$ is a project graph at temporal resolution, t_i and for each Projected graph $PG_{(t_i)}$.
- Step 5: Compute graph metrics, M_i, where M_i is a graph metric (e.g. graph Density, Volume and Avg. path length) between consecutive M_i s.
- Step 6: Compare and correlate consecutive M_i values and compute delta Δ.
- Step 7: Identify and analyse evolutionary patterns in Δ values.

These steps allow a user to define y time-window within the life-time of the Whole-graph where the evolutionary analysis is required. Subgraphs of interest can also be defined such that the evolutionary analysis involves only one or more subgraphs in the space dimension of the network. After defining the time-windows and subgraphs of interest, the temporal resolution (such as seconds, minutes, hours, days, months and years) is defined. This is temporal resolution at which the Whole-graph is projected, and the evolution is captured using selected graph metrics. These steps are captured in an algorithm described in Table 1.

The algorithmic workflow can be implemented in a graph query to project the state of the Whole-graph and compute the graph metrics at varying temporal resolutions within any given time-window. Time-series analysis methods can be used thereafter to analyze the computed graph metrics values across time for evolutionary behavior of the graph [33]. For example, if we have the graph Density at time t_i as Dt_i, at t_j time Dt_j and it

Table 1 Capturing the Whole graph evolution algorithm

1. Define a time-window, $\Upsilon \subset T$ for analysis
2. (where Υ is the selected time-window
3. and T is the entire lifetime of the whole graph)
4. Define subgraph(s) of interest, $S^{'} \subset S$
5. (where $S^{'}$ is the selected subgraph (s)
6. and S is the entire space-dimension of the whole graph)
7. Define the temporal resolution $t_i \subset \Upsilon$ for iterative projections
8. FOREACH t_i in Υ
9. Project Graph, $PG_{(t_i)}$
10. (where $PG_{(t_i)}$ is a project graph at temporal resolution,t_i)
11. FOREACH Graph $PG_{(t_i)}$
12. Compute Graph Metrics, M_i
13. (where M_i is a graph metric (e.g. graph Density, Volume and Avg. path length)
14. FOREACH Consecutive M_i
15. Compare M_i values, delta Δ
16. Identify patterns in Δ values

continually evolves throughout the entire graph life-time, $(\mathrm{D}t_i, \ldots, \mathrm{D}t_n)$, we can use time series analysis method to depict the evolution of the graph's density at different temporal resolutions.

Graph metrics

Graph metrics such as network density, network diameter and average path length have been utilized in this case study to retrieve the evolutionary pattern of the transit network structure at different temporal resolutions over monthly and yearly time-windows.

Graph density

One of the most commonly used metrics for observing evolution in network structure is the graph density. In this experimental study, graph density of each Projected graph $PG_{(t_i)}$ in the STVG measures how close a given $PG_{(t_i)}$ is to a complete graph. That is, the ratio between the number of its edges and the number of all possible edges between n nodes.

$$\text{Graph density D} = \frac{|E|}{|N| * (|N| - 1)}$$

However, in real practical sense, true graph density ρ(G) = Mass(G)/Volume (G), where the Mass of the graph G is a total mass of its edges and nodes, and Volume (G) is a size-like graph characteristic of the graph [34].

Mass(G) = $|E|$ which approximately estimates the Mass of the graph while;

Volume (G), = $|N| * (|N| - 1)$ which approximately estimates the Volume (Size) of the graph which is dependent on the number of nodes. The evolution of the density of the whole-graph can be observed by computing and observing the trend over the Projected graph $PG_{(t_i - t_n)}$ through time, that is

$$Density = \frac{E_{(t_i - t_n)}}{\left|N_{(t_i - t_n)}\right| * \left(\left|N_{(t_i - t_n)}\right| - 1\right)}.$$

Average path length

Average path length otherwise known as characteristic path length is also one of most robust measures of network topology [34]. It is denoted by ⟨d⟩ and can be used as a measure of the efficiency of travel or mass transit efficiency in the network. For each PG_i, Average Path Length can be measured as:

$$\langle \mathrm{d} \rangle = \frac{1}{N(N-1)} \sum_{i<>j,N} d_{i,j},$$

where d_{ij} is the number of edges for the shortest path from node i to node j. And the evolution of ⟨d⟩ can be observed through PG_i over time $(ti - tn)$; where the ⟨d⟩ *of* $PG_{(ti-tn)}$ can be defined as

$$\frac{1}{\left|N_{(t_i-t_n)}\right| * \left(\left|N_{(t_i-t_n)}\right| - 1\right)} \sum_{i<>j,N} d_{i,j}(t_i - t_n)$$

Experimental study

In this section, we describe an experimental study and implementation, utilizing the STVG framework to carry out evolutionary graph analytics of a bus transit network.

Data description

The datasets used for the implementation are the Automatic Vehicle locations (AVL) feeds from the CODIAC bus transit network in Greater Moncton, New Brunswick, Canada and GIS shapefile dataset containing information about the bus routes, bus stops, street segments, street intersections, civic addresses obtained from GeoNB online service (http://www.snb.ca/geonb1/e/index-E.asp). We extracted a total of 59,617 completed bus trips from the AVL feeds pulled from the buses every 5 s into a PostgreSQL/Post GIS database for a period of 18 months (from June 2016 -December 2017). The raw data set from the AVL feeds consist of the following attributes:

- vlr id: The ID of the data point in the vehicle location reports table.
- route id vlr: The route ID in the vehicle location reports table.
- route name: The route name.
- route id rta: The route ID in the route transit authority table.
- route nickname: The abbreviate of the route.
- trip id br: The trip ID in the bid route table.
- transit authority service time id: Transit authority service time ID.
- trip id tta: Transit authority trip ID.
- trip start: Start time of the trip.
- trip finish: Finish time of the trip.
- lat: Latitude.
- lng: Longitude.
- timestamp.

Automated pre-processing tasks

The data pre-processing tasks involve the steps that have been used to transform the raw datasets into a suitable format for building the STVG. These tasks were implemented by an automated data pre-processing algorithm developed by [35] which includes the following steps:

Step 1: Transformation of GPS points into moves/stops The algorithm utilizes a fixed distance value between two consecutive points to determine if the bus was moving, "*Moves*" or stopped, "*Stops*". This fixed distance was empirically determined as being 15 m for a transit network. If the distance between the previous point and the current point is more than 15 m, the current point is annotated as a "*Move*". In contrast, if the distance is less than 15 m, the current point is annotated as a "Stop". These distances are Euclidian distances; network distance could be considered but for a short distance of 15 m, Euclidian distance can be used efficiently for this purpose.

Step 2: Street name annotation The algorithm annotates the "*Moves*" and "*Stops*" computed from the previous step with the street names if the event was performed on a street. For this task, a query on a PostgreSQL database is run to retrieve the names of the streets where a "*Move*" or "*Stop*" is located. This is a non-trivial step because the GPS coordinates usually have 10 m of accuracy in urban areas [36].

Step 3: Bus stop identification This step annotates the "*Moves*" and "*Stops*" with the "*Bus Stop*" names if the event took place within a bus stop in a 30 m buffer zone. It is important to point out that the algorithm also needs to verify the direction of a moving bus (e.g. eastbound and westbound) to identify the "Bus *Stop*" that a "*Stop*" or "Move" is located at.

Step 4: Street intersection identification This step also annotates the "*Moves*" and "*Stops*" with the "*Street intersection*" names if this event took place within a street intersection. To determine if the mobility action is within a street intersection, the algorithm creates a buffer zone of a 30 m radius (determined empirically) for each street intersection. The "*Stops*" and *Moves*" that are located inside a given buffer zone are annotated with the intersection identifier.

Step 5: Origin/destination trip identification The algorithm in this step identifies the origin and destination of each trip. The first GPS point of a bus trip located at a bus stop or station is tagged as the origin, and the last point of that same trip ID located at a bus stop or station is tagged as the destination. The GPS points between the origin and destinations points are sequentially indexed in order of occurrence.

Step 6: Bus trip labeling The algorithm labels the trips in terms of (Route Number-Run Number-Run-day-Run Month-Run Year), where (50-10-23-12-2016) would represent Route #50 during the 10th run of the 23rd day of 12th month of 2016. The date had to be concatenated to the trip IDs, so they could have unique IDs in the database. A *"Trip"* here is a completed journey of a bus from an origin to a final destination.

At the end of this preprocessing pipeline, a CSV file is automatically generated containing all the data needed to build the STVG of the bus transit network in Neo4j graph database.

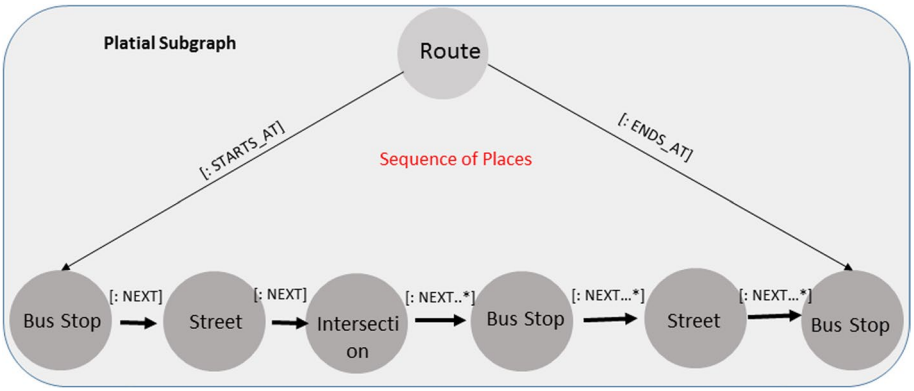


Fig. 3 Platial subgraph

Building the subgraphs

We define the Platial and Mobility subgraphs for generating the Whole-graph of the bus transit network which represents the connectivity between mobility events and places over time. The pre-processed data sets in the CSV file are used to build the subgraphs of the bus transit network which generates a Whole-graph. The smallest temporal resolution at which nodes and edges were inserted into the subgraphs was 5 s, corresponding to the time-interval of the AVL feeds. This means changes in the subgraphs can be tracked to the detail of every 5 s. We utilized the Neo4j graph database for the implementation. Neo4j is currently the most popular native graph database widely used for graph data management and analytics. The language in building and processing the graph in Neo4j is called Cypher, which we have used to write the nodes and edges continuously into the graph and used for encoding graph metrics for evolutionary graph analytics.

Platial subgraph

The Platial subgraph consists of a sequence of physical geographical places that represent a "*Trip*" of the bus within the transit network. In this case, a "*Trip*" is represented as a bus route that consists of a static "*Bus Stop*" (i.e. a designated place for passengers to board or alight from a bus), "*Street Segment*" (i.e. the transit segment between two bus stops belonging to a bus route), and "*Street Intersection*" (e.g. existing 3-way intersections or 4-way intersections of a bus route). More geographical places can be defined depending on the availability of dataset and application. In this case study, nodes are created in the Platial subgraph according to the actual locations of the bus trips at any given time. The "NEXT" edges represent adjacency relationships between two consecutive geographical places in a trip. Figure 3 illustrates the network structure of the Platial subgraph in the STVG of the bus transit network. A node in the platial subgraph can reoccur in more than one Projected graph even in tumbling time-windows.

The Cypher query as shown in Table 2 was used to import the bus stop, intersection street segments points as Bus stop nodes, Intersection nodes and Street segment nodes

Table 2 Cypher query used to create Bus Stop nodes in the Platial Subgraph

```
1. Load csv with headers from
2. 'file:///busstops.csv' as csv
3. Create (bs:BusStop {BusStopID: csv.stop_id, sName: csv.stop_name, sLat:
4. toFloat(csv.stop_lat), sLon: toFloat(csv.stop_lon),
5. sParentStation: csv.parent_station, locType: csv.location_type});
```

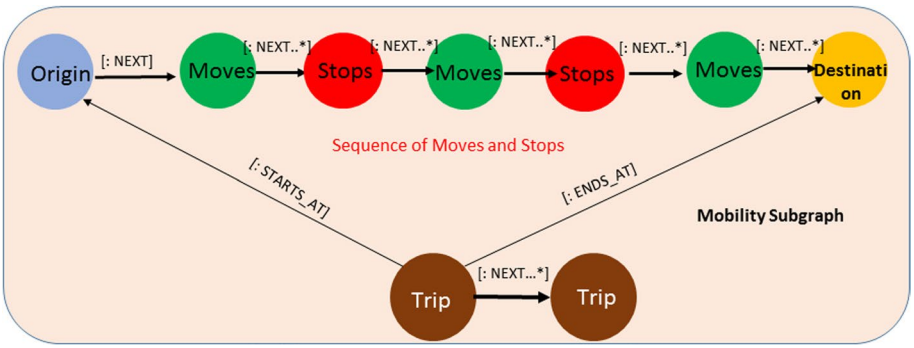


Fig. 4 Mobility subgraph

respectively to the graph database. This step also includes the nodes properties such as intersection ID, intersection name and street segment ID, and street segment name.

Mobility subgraph

The Mobility subgraph represents the discrete sequence of mobility events (moves and stops) of a moving bus in space and time. In this subgraph, as shown in Fig. 4, the primary entity is the "*Trip*" that represents the trajectory of a moving bus within a bus route of a transit network. A *"Trip"* node is a composite node which is created by the sequence of "Move" (i.e. a node representing the location where a bus is in motion), and "Stop" (i.e. a node representing the location where the bus is not in motion), nodes from an "*Origin*" (i.e. a node representing the first location of a trip) to a "*Destination*" (i.e. a node representing the last location of a trip). The connectivity between these nodes is represented by the "NEXT" adjacency relationship which is a space–time relation such as *Origin-Move*, *Stop–Stop, Stop-Move, Stop-Destination, Move-Destination,* and *Move-Stop.*

Each node in the Mobility subgraph occurs at a specific point in time and adds a new node and their respective NEXT relation to the graph. These nodes are connected to the time-tree's leaf nodes using their date and timestamps, starting from the year to the second leaf nodes. This is important for tracking the evolution of the subgraph. The "Origin" and "Destination" nodes of a trip have relationships to the "Trip" node labelled as "START_AT" and "ENDS_AT" respectively which gives meaning to the edges and useful for the evolutionary analysis of the trips. This subgraph is majorly responsible for the dynamics of the graph. In this case study, every 5 s, a node or more from the mobility subgraph is added to the whole-graph.

Table 3 Cypher query used to create "Moves" and "Stops" nodes

"Moves" nodes	"Stops" nodes
Load csv with headers from	*Load csv with headers from*
'file:///data/Moves.csv' as csv	*'file:///data/Stops.csv' as csv*
MERGE(r:Moves{MoveID: csv.moveid})	*MERGE (r:Stops {StopID: csv.stopid}) ON CREATE SET*
ON CREATE SET	*r.TripID = csv.Tripid, r.Street = csv.streetname,*
r.TripID = csv.Tripid, r.Street = csv.streetname,	*r.BusStop = csv.BusStop,r.latitude = toFloat(csv.lat),*
r.BusStop = csv.Busstop,	*r.longitude = toFloat(csv.long),*
r.latitude = toFloat(csv.lat), r.longitude = toFloat(csv.long),	*r.Date = csv.Date, r.Time = csv.Time,*
r.Date = csv.Date,r.Time = csv.Time, r.Sequence = toInteger(csv.sequence),r.State = csv.state,	*r.Sequence = toInteger(csv.sequence),r.State = csv.state, r.year = toInteger(substring(csv.Date,6,4)),*
r.year = toInteger(substring(csv.Date,6,4)),	*r.month = toInteger(substring(csv.Date,3,2)),*
r.month = toInteger(substring(csv.Date,3,2)),r. day = toInteger(substring(csv.Date,0,2)),	*r.day = toInteger(substring(csv. Date,0,2)),*
r.hour = toInteger(substring(csv.Time,0,2));	*r.hour = toInteger(substring(csv.Time,0,2));*

The Mobility subgraph nodes were continuously loaded while concurrently establishing the connectivity with the Platial subgraph nodes. The cypher statements for importing the "Moves" and "Stops" nodes in the whole-graph are shown in Table 3.

Connectivity between mobility and platial subgraphs

Ideally, each node in the Mobility subgraph has a corresponding node in the Platial subgraph where it is located at in a given time instance. Platial and Mobility subgraph nodes are complementary in space and time (Fig. 5). They are complementary because for each node in the Mobility subgraph there is always a corresponding node in the Platial subgraph on which a connection is established using a cypher statement as shown in Table 4.

The connectivity between the two subgraphs is semantically represented as the "LOCATED_AT" edges that exists between the corresponding nodes. Independent of the number of subgraphs being represented in the graph model, all nodes are connected to a time-graph based on the temporal relationship between the nodes in only one of the subgraphs and the leaf nodes of the time-graph. The Mobility subgraph nodes are connected to the time-graph from top to the lowest level of temporal resolutions using, for example the Cypher statement in Table 5 to create the sequential "HAPPEN_AT" edges between the "*Move*" nodes and their corresponding time-graph leaf nodes.

The time-graph is the time-dimension of the Whole-graph which contains all possible time instances of every node in the subgraphs throughout the graph lifetime. We built a time-graph of a 2-year lifetime (2016–2017) in hierarchical and sequential order of temporal resolutions as described in the Cypher statements in Table 6.

The Whole-graph

The whole-graph is composed of the subgraphs and the time-graph. It presents the complete connectivity between the space and time dimensions of the network. Figure 6 represents the overview of two dimensions of the graph model, the space dimension, which consists of the subgraphs, and the time dimensions that consists of the time-graph. The connectivity between these dimensions enables the evolutionary analytics of the transit

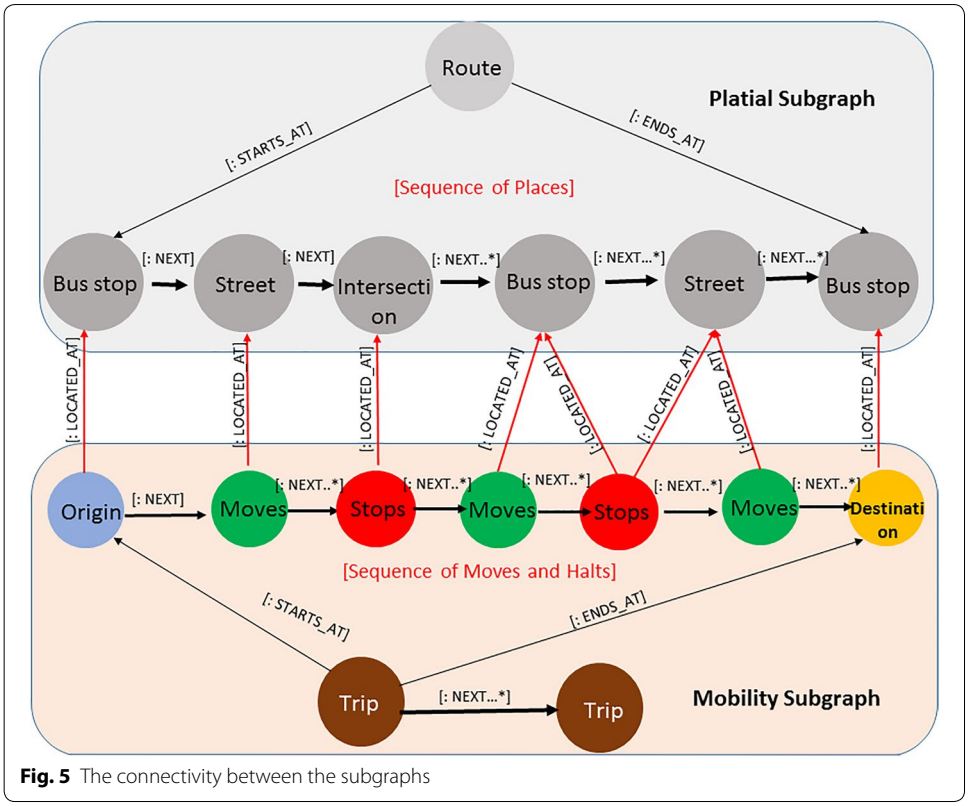


Fig. 5 The connectivity between the subgraphs

Table 4 Cypher query used to create the edges between mobility and platial subgraphs

```
MATCH (bs:BusStop), (st:Stops) WHERE bs.BusStopID = st.Busstop
MERGE (st)-[:LOCATED_AT]->(bs);
MATCH (ss:Streets), (st:Stops) WHERE ss.STNoSpace = st.Street
MERGE (st)-[: LOCATED_AT]->(ss);
MATCH (ss:Streets), (st:Moves) WHERE ss.STNoSpace = st.Street
MERGE (st)-[: LOCATED_AT]->(ss);
MATCH (bs:BusStop), (st:Moves) WHERE bs.BusStopID = st.Busstop
MERGE (st)-[: LOCATED_AT]->(bs);
```

Table 5 Sequential connection of the moves nodes to the time-graph

```
MATCH (t:Moves) WITH t
MATCH (yy:Year {yearid:t.year}) WITH t,yy
MATCH (yy)-[r1]- > (mm:Month {monthid:t.month}) WITH t,yy,mm
MATCH (mm)-[r2]- > (dd:Day {dayid:t.day}) WITH t,yy,mm,dd
MATCH (dd)-[r3]- > (hh:Hour {hourid:t.hour}) WITH t,yy,mm,dd, hh
MATCH (hh)-[r4]- > (mm1:Minute {minuteid:t.minute})WITHt,yy,mm,dd,hh, mm1
MATCH(mm1)-[r5]- > (ss:Second{secondid:t.second})WITH t,yy,mm,dd,hh,mm1,ss
CREATE (t)-[:HAPPENS_AT]- > (ss);
```

Table 6 Cypher query for creating the time-graph

```
1. WITH range(2016, 2017) AS YEARS, range(1-12) as MONTHS, range() as Days, range(1,24) as Hours, range(1,60) as Minutes,
2. range(1,60) as Seconds
3. FOREACH(year IN YEARS | MERGE (y:Year {yearid: year})
4.     FOREACH( month IN MONTHS | CREATE (m:Month {monthid: month})
5.         MERGE (y)-[:CONTAINS]->(m)
6.         FOREACH(day IN (CASE
7.                         WHEN month IN [1,3,5,7,8,10,12] THEN range(1,31)
8.                         WHEN month = 2 THEN
9.                             CASE
10.                                WHEN year % 4 <> 0 THEN range(1,28)
11.                                WHEN year % 100 = 0 AND year % 400 = 0 THEN range(1,29)
12.                                ELSE range(1,28)
13.                            END
14.                        ELSE range(1,30)
15.                    END) |
16. CREATE (d:Day {value: day}
17. MERGE (m)-[:CONTAINS]->(d)
18. FOREACH( hour IN Hours | CREATE (h:Hour {hourid:hour})
19.  MERGE (d)-[:CONTAINS]->(h)
20.     FOREACH ( minute in Minutes | CREATE (mm:Minute {minuteid:minute})
21.         MERGE (mm)<-[:CONTAINS]-(h)
22.             FOREACH ( seconds in Seconds | CREATE (ss:Second{secondid:second})
23.                 MERGE (mm)<-[:CONTAINS]-(mm)) ))) );
```

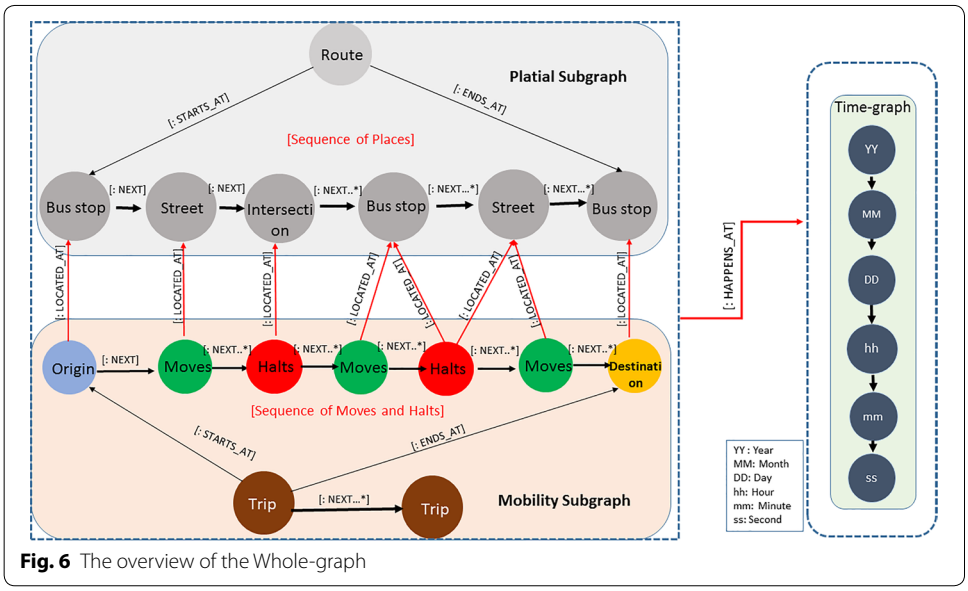
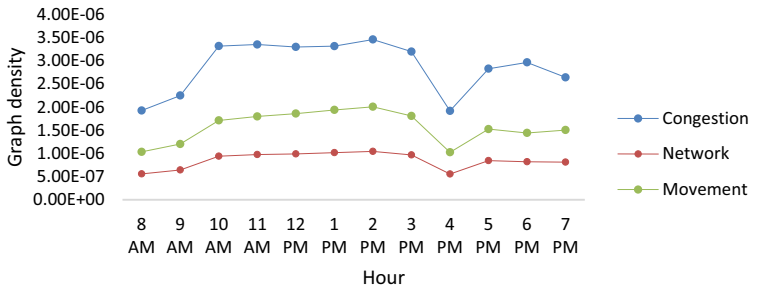


Fig. 6 The overview of the Whole-graph

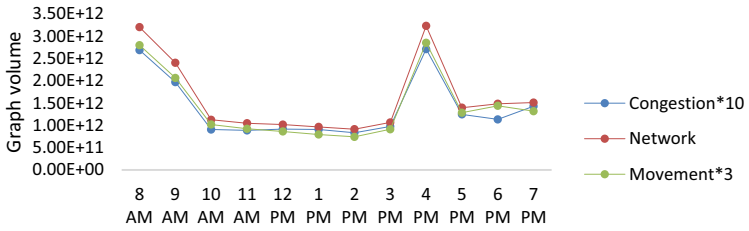
Table 7 Statistics of the Whole-graph

Nodes	153,127,231
Edges	453,713,224
Average clustering coefficient	0.1541
Number of triangles	731,205,512
Diameter (longest shortest path)	659
Time span	548 days

network graph. The time-graph does not only track the evolution of the space dimension, but it also facilitates efficient retrieval of time-dependent Projected graphs. Table 7 describes the statistics of the Whole-graph in the graph database.



a *Evolution of graph density at hourly temporal resolution.*



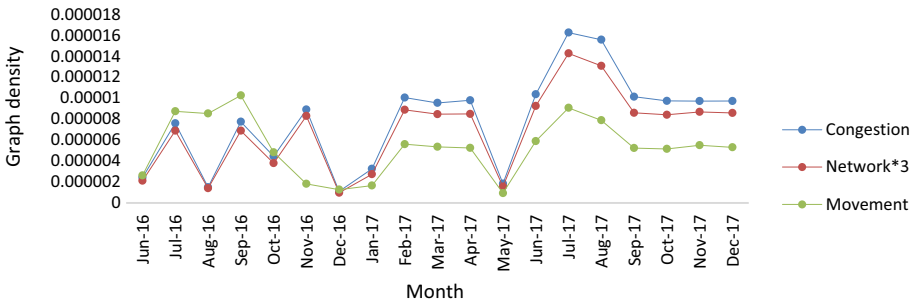
b *Evolution of graph volume at hourly temporal resolution.*

Fig. 7 Evolution of Graph density and volume at hourly temporal resolutions

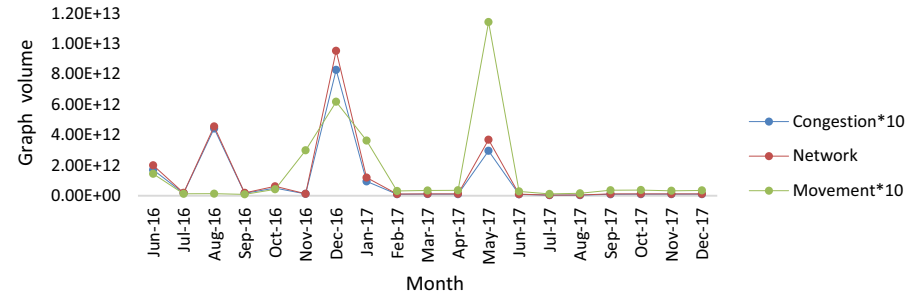
Results and discussion

From a global perspective, the trend in the graph density and volume as shown in Fig. 7a, b reflect the bus transit network's topology variations within a 2-year time window at hourly temporal resolution. This trend could be useful in observing increase or decrease in bus transit mobility activities over time, as well as trip density within the entire network through time. We could also observe if there is a significant correlation between global network density and congestion density in the network through time. Congestions in our graph model represents where there are many of "Stops" on a street with a "Move".

Figure 7a also reveals an evolutionary trend in the Whole-graph of the transit network such that the hourly network density, graph density with respect to traffic congestion and bus movement have the lowest values at peak transit hours (8 a.m. and 4 p.m.). In a practical sense, the trend reveals the highest volume of transit mobility activities ("Moves" and "Stops"), trips and traffic congestion at these peak hours. In other words, between 8 am and 9 am in the mornings as well as 4 p.m. and 5 p.m. in the evenings, more nodes and edges are added to the Whole-graph of the transit network signifying a lot of mobility activities within the network during these times. The inverse relationship between graph density and graph volume is clearly depicted in Fig. 8a, b where an increase in volume means decrease in density and vice versa. The evolutionary trend shows a strong correlation between the global network density of the graph, graph density with respect to traffic congestion and density with respect to the bus movement. Similar behavior is seen in the network volume, congestion and movement volume, as one increases, others increase as well.



a *Evolution of graph density at monthly temporal resolution.*



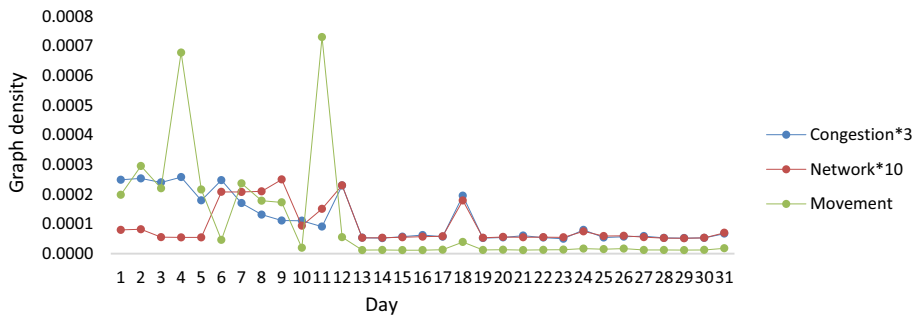
b *Evolution of graph volume at monthly temporal resolution.*

Fig. 8 Evolution of graph density and volume at monthly temporal resolutions

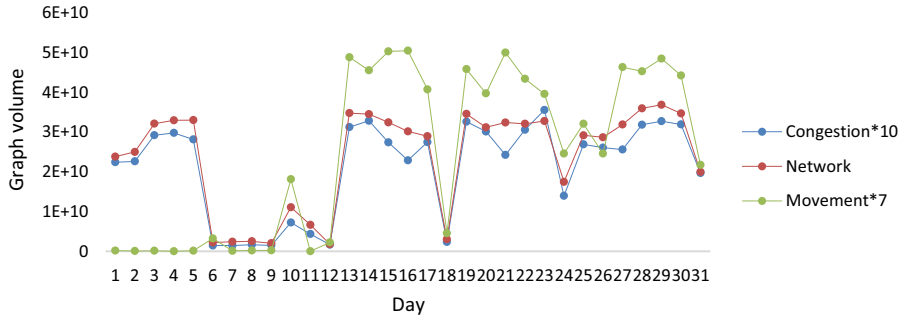
In monthly temporal resolution, Fig. 8a, b reveal an evolutionary behavior with peaks in August 2016, December 2016 and May 2017 in terms of movement, traffic congestion and network volume as well as density of the graph.

From June to December 2016, graph density with respect to the entire network, movement and congestion reveal a zig-zag (up and down) alternate behavior different from what is observed in the same months in 2017. The graph volume with respect to congestion, movement as well as that of the entire network from June to December 2017, did not experience significant changes. The peaks observed in August 2016 and December 2016 did not reoccur in the following year, 2017. Figure 9a, b provides a closer look at the behavior of the transit network in a monthly time-window (December 2016 peak) on a daily temporal resolution. The daily behavior of transit activity in December 2016 is compared to that of the previous month, November 2016 to depict the differences in daily transit pattern and the reason for such a peak in December.

The daily graph volume and density behavior for the December 2016 bus transit network activity peak depicts high volume of bus transit activity in many days of the month except the lows observed between Tuesday 6th to Friday 9th as well as on Monday 12th. There is also the weekend low such as on Sunday 18th, Saturday 24th and 31st of December 2016. Comparing December 2016 daily network volume with that of the previous month, November 2016, as shown in Fig. 10, we could deduce the reason for the December peak. There is more transit network low volume in November with respect to that of December. From 12th of December to 30th, there are more mobility events in the network when compared to those of November. In graph topological sense, more nodes and edges are added to the whole-graph in December than in November 2016.



a *Evolution of graph density at daily temporal resolutions for December 2016.*



b *Evolution of graph volume at daily temporal resolutions for December 2016*

Fig. 9 Evolution of graph density and volume at daily temporal resolutions for December 2016

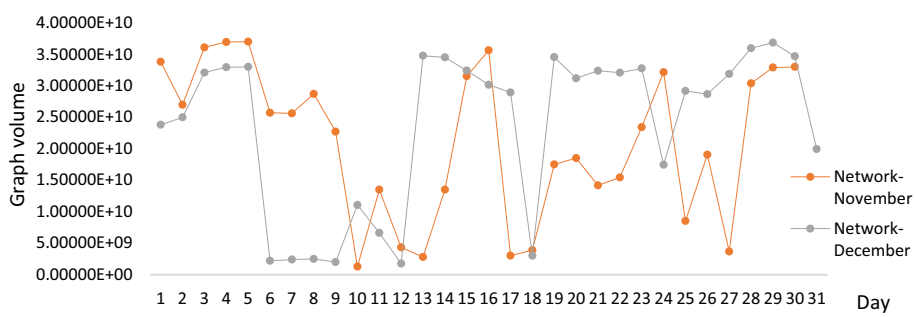
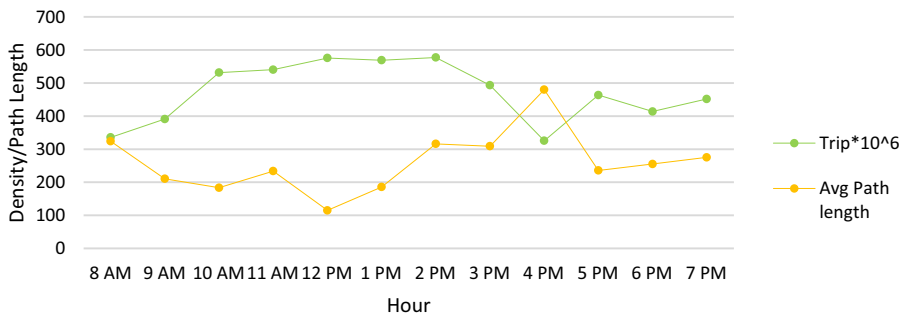


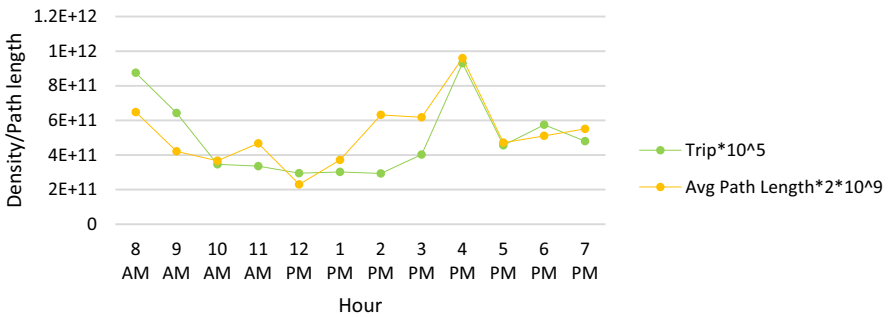
Fig. 10 Evolution of network volume at daily temporal resolution for November and December 2016

The evolutionary trend in the Average path length of the network would reflect the average trip length (travel time) over time. One analysis of interest would be to observe how evolving average path length correlates with evolving trip density/volume over time in the network.

It can be observed in Fig. 11a, b, that Average path length has a very strong correlation with graph density and volume. It increases with increase in graph volume and decreases with an increase in graph density. In a practical sense, average path length in this case study is the average trip length (travel time) from an origin to destination within the network. It is as expected that travel time (trip length) would increase when volume of congestion and mobility activity in the network increase, as observed in Fig. 11a. We can



a *Evolution of avg. path length vis-a-vis graph density at hourly time-windows.*



b *Evolution of avg. path length vis-a-vis graph volume at hourly time-windows*

Fig. 11 Evolution of avg. path length vis-a-vis graph density and volume at hourly time-windows

also observe in Fig. 11b that the highest average trip lengths are seen at peak hours of the day (8am and 4 pm) when it shows the highest traffic congestion and network volume.

Conclusions

Evolutionary graph analytics based on sequence of graph snapshots have been commonly utilized in literature mainly because of the convenience it affords the users to manage small-size graph in discrete versions and compare differences between snapshots across consecutive time windows as this approach may be suitable for small graphs and for cases where changes do not occur at shorter temporal resolutions. It is associated with high storage overhead in proportion to the size of the evolving graph and the time intervals between snapshots because the entire graph is usually replicated from one snapshot to another. Also, computing evolving queries across the snapshots is computationally intensive and complicated. These are major disadvantages especially, in cases where the graph is massive and changes frequently at shorter temporal resolutions and where the evolutionary analysis relies on the explanatory power of representing the dynamics across different temporal resolutions.

We therefore, propose a framework based on our Space–Time-varying graph (STVG) formalism which utilizes the Whole-graph approach to model the dynamics of the graph whose evolutions materialize in the time-varying changes in its Projected graphs. The STVG framework provides an approach to reduce high storage overhead of massively changing graphs where new nodes and edges arrive every second. It affords the capability to extract Projected graphs at different time-windows and analyze their metrics across varying temporal resolutions.

The framework was implemented for a transit case study using the AVL feeds of the bus transit network of Greater Moncton, New Brunswick, Canada which generated a Whole-graph of 44.2 GB in the database. In contrast, using the Snapshot method created 7280 hourly snapshots, 732 daily snapshots, 18 monthly snapshots and 2 yearly snapshots in the graph database where the smallest snapshot amounted to 1.3 GB of graph. The total storage cost for over 18,000 snapshots of graph needed for our evolutionary graph analytics amounted to 23.4 TB, because the total storage and computational cost increase linearly with the number of snapshots, that is, total cost is equal to, cost per snapshot multiplied by the number of snapshots. Using the Snapshot method presents over 500 times increase in storage overhead when compared to our Whole-graph approach.

Our evolutionary analysis was based on graph density, volume and average path length on the Projected graphs at varying temporal resolution across different time windows. The analysis reveals evolutionary patterns in the overall network density of the graph, traffic congestion density as well as graph density with respect to bus movement at hourly, daily and monthly temporal resolutions. Similar patterns are observed in the evolutionary pattern of network volume, congestion and movement volumes. This type of analytics for any transit networks potentially provides an efficient way to uncover dynamics of the network as well as the dynamics in the network over space and time. The evolutionary pattern of the transit network properties such as average paths, network density and volume as a result of the dynamics in human mobility pattern may become vital for transit network optimization. Potential applications can be found in transit trend analysis as well as in time-dependent transit recommendation systems.

Abbreviations
AVL: automatic vehicle; CSV: comma separated values; GIS: geographic information systems; GPS: global positioning system; ID: identity; NB: New Brunswick; PG: Projected graphs; STVG: Space–Time-varying graph.

Acknowledgements
We want to acknowledge CODIAC for giving us access to the dataset.

Authors' contributions
MW and IM developed the conceptual framework. IM carried out every implementation. All the authors contributed in the writing of the paper. All authors read and approved the final manuscript.

Funding
This research was supported by TETFund and NSERC/Cisco Industrial Research Chair.

Availability of data and materials
The data is not available to public because CODIAC bus transit agency allowed us to use the data on the condition that the data shall not be distributed or shared.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] People in Motion Lab, University of New Brunswick, 15 Dineen Drive, Fredericton, NB E3B 5A3, Canada. [2] Civil Engineering, University of New Brunswick, 15 Dineen Drive, Fredericton, NB E3B 5A3, Canada.

Received: 27 March 2019 Accepted: 12 June 2019
Published online: 21 June 2019

References
1. Huang Q, Zhao C, Zhang X, Wang X, Yi D. Centrality measures in temporal networks with time series analysis. Europhys Lett. 2017;118(3):36001.
2. Khurana U, Deshpande A. Storing and analyzing historical graph data at scale. 2016; 65–76.

3. Khurana U. An introduction to temporal graph data management. 2016. http://citeseerx.ist.psu.edu. p. 1–11.
4. Tong H, Papadimitriou S, Yu PS, Faloutsos C. Proximity tracking on dynamic bipartite graphs: problem definitions and fast solutions. In: Tong H, Papadimitriou S, Yu PS, Faloutsos C, editors. Link mining: models, algorithms, and applications. New York: Springer; 2010. p. 211–36.
5. Rossi RA, Gallagher B, Neville J, Henderson K. Modeling dynamic behavior in large evolving graphs. In: Proc. sixth ACM Int. Conf. Web search data Min.—WSDM'13. 2013. p. 667.
6. Yang Y, Yu JX, Gao H, Pei J, Li J. Mining most frequently changing component in evolving graphs. World Wide Web. 2014;17(3):351–76.
7. Yang J, Leskovec J. Patterns of temporal variation in online media. In: Proc. fourth ACM Int. Conf. Web search data Min.—WSDM'11. 2011. p. 177.
8. Pereira FSF, de Amo S, Gama J. Evolving centralities in temporal graphs: a twitter network analysis. In: 2016 17th IEEE Int. Conf. Mob. Data Manag. 2016. p. 43–48.
9. Koloniari G, Pitoura E. Partial view selection for evolving social graphs. In: GRADES'13 first international workshop on graph data management experiences and systems. 2013. p. 4503–2188.
10. Aridhi S, Montresor A, Velegrakis Y. BLADYG: a graph processing framework for large dynamic graphs. Big Data Res. 2017;9:9–17.
11. Leventhal GE, Hill AL, Nowak MA, Bonhoeffer S. Evolution and emergence of infectious diseases in theoretical and real-world networks. Nat Commun. 2015;6:6101.
12. Magnien C, Tarissan F. Time evolution of the importance of nodes in dynamic networks. In: IEEE/ACM international conference on advances in social networks analysis and mining. 2015.
13. Starnini M, Machens A, Cattuto C, Barrat A, Pastor-Satorras R. Immunization strategies for epidemic processes in time-varying contact networks. J Theor Biol. 2013;337:89–100.
14. Jalili M, Salehzadeh-Yazdi A, Gupta S, Wolkenhauer O, Yaghmaie M, Resendis-Antonio O, Alimoghaddam K. Evolution of centrality measurements for the detection of essential proteins in biological networks. Front. Physiol. 2016;7:375.
15. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp. 2008;10:2008.
16. Yarlagadda R, Pinnaka S, Etinkaya EKÇ. A Time-evolving weighted-graph analysis of global petroleum exchange. In: 7th international workshop on reliable networks design and modeling (RNDM), 2015.
17. Von Landesberger T, Brodkorb F, Roskosch P, Andrienko N, Andrienko G, Kerren A, Member S. Mobility graphs: visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. IEEE Trans Vis Comput Graph. 2015;22:11–20.
18. Qi X, Liu L, Cai G, Xie M. A topology evolution model based on revised PageRank algorithm and node importance for wireless sensor networks. Hindawi. 2015.
19. Glacet C, Fiore M, Gramaglia M. Temporal connectivity of vehicular networks: The power of store-carry-and-forward. In: IEEE Vehicular Networking Conference, VNC. 2016.
20. Küçükkeçeci C, Yazıcı A. Big data model simulation on a graph database for surveillance in wireless multimedia sensor networks. Big Data Res. 2018;11:33–43.
21. Aggarwal C, Subbian K. Evolutionary network analysis: a survey. ACM Comput Surv. 2014;47:10.
22. Ivarsson T, Kollegger A, Neubauer P, Svensson J, Webber J. The Neo4j manual v1. 3. USA: Neo-Technology. 2014.
23. Casteigts A, Flocchini P, Quattrociocchi W, Santoro N. Time-Varying graphs and dynamic networks. Ad hoc Mob Wirel Netw. 2011;6811:346–59.
24. Gottumukkala RN, Venna SR, Raghavan V. Visual analytics of time evolving large-scale graphs. IEEE Intell Inf Bull. 2015;16(1):10–6.
25. Huo W, Tsotras VJ. Efficient temporal shortest path queries on evolving social graphs. In: Proceedings of the 26th international conference on scientific and statistical database management—SSDBM'14. 2014. p. 1–4.
26. Ren C, Lo E, Kao B, Zhu X, Cheng R. On Querying Historical Evolving Graph Sequences. Proc VLDB Endow. 2011;4(11):726–37.
27. Lerman K, Ghosh R, Kang JH. Centrality Metric for Dynamic Networks. Inf. Sci. (Ny). 2010;354(Pt 3):70–7.
28. Quattrociocchi W, Amblard F, Galeota E. Selection in scientific networks. Soc Netw Anal Min. 2012;2(3):229–37.
29. Kumar R, Novak J, Tomkins A. Structure and evolution of online social networks. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining—KDD'06. 2006. p. 611.
30. Liu Z, Yu JX. Discovering burst areas in fast evolving graphs. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2010, vol. 5981 LNCS, no. PART 1, p. 171–185.
31. Santoro N, Quattrociocchi W, Flocchini P, Casteigts A, Amblard F. Time-varying graphs and social network analysis : temporal indicators and metrics. In: 3rd AISB Soc. networks multiagent Syst. Symp. 2011. p. 32–38.
32. Ferreira A. Building a reference combinatorial model for MANETs. IEEE Netw. 2004;18(5):24–9.
33. Fenn DJ, Porter MA, Williams S, McDonald M, Johnson NF, Jones NS. Temporal evolution of financial-market correlations. Phys Rev E Stat Nonlinear Soft Matter Phys. 2011;84(2):1–13.
34. Edwards B, Hofmeyr S, Stelle G, Forrest S. Internet topology over time. 2012. Arxiv, p. 6.
35. Cao H, Wachowicz M. The design of an IoT-GIS platform for performing automated analytical tasks. Comput Environ Urban Syst. 2019;74:23–40.
36. Salarian M, Manavella A, Ansari R. Accurate localization in dense urban area using Google street view images. In: 2015 SAI intelligent systems conference (IntelliSys), 2015. p. 485–490.

Publisher's Note