Evaluate Topological Quality of Converted Network Model in Volunteered Geographic Information

Tsz-kin CHAN

BSc (Hons.) Geomatics

The Hong Kong Polytechnic University

2019

(This page is intentionally left blank.)

The Hong Kong Polytechnic University

The Department of Land Surveying and Geo-Informatics

Evaluate Topological Quality of Converted Network Model in Volunteered Geographic Information

Tsz-kin CHAN

A dissertation submitted in partial fulfilment of the requirements for the degree of

BSc (Hons.) Geomatics

CERTIFICATE OF ORIGINALITY

I hereby declare that this dissertation is my work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

(Signature)

Tsz-kin CHAN

(Name)

2019/07/23

(Date)

ABSTRACT

Abstract of dissertation entitled:

Evaluate Topological Quality of Converted Network Model in Volunteered Geographic Information

Submitted by Tsz-kin, CHAN

for the degree of Bachelor of Science (Honours) Geomatics

at The Hong Kong Polytechnic University

in July 2019.

Hong Kong has been promoting the Smart City initiatives recently. One major task for the authorities is to let the residents "walk smartly" by providing useful information on the routes. However, the underlying information supported the smart direction, the network format, have not been published. It is a natural response to find an alternative solution on network format.

Meanwhile, Volunteered Geographic Information has become a hot topic since one decade ago. Recently an application has been developed to extract network model from them. Past studies have been done on the data quality on the Volunteered Geographic Information. Interestingly, the investigations have not studied how well the network, created from those data, perform versus the ground truth. The worse scenario of the problem is obsolete of much network-based application of the Volunteered Geographic Information.

This dissertation designed an algorithm to evaluate the topological quality of Volunteered Geographic Information versus a reference network. With the data and the application, the difference between the structures of the two networks could be visualised in both numerical and graphical format. A case study in East Tsim Sha Tsui, Hong Kong has been conducted in order to implement the algorithm. The major findings on the study area show that those public co-creation products have flaws in their network structures compared to a reference. Although analysis on the main skeleteon would be still acceptable, the network structure is not accurate enough for tasks requires complete accuracy.

ACKNOWLEDGEMENT

I want to express sincere gratitude to Dr Xintao Liu, the supervisor of this dissertation at the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University. His support on this project and consideration of my health realise this dissertation.

Dr Christopher Donald Higgins would be the second most grateful person for this project. This dissertation cannot be draft without his offer in last fall — all the best in Toronto.

Moreover, it was a memorial time with friends at university. The pleasurable time with them would not be forgotten. In particular: June Ng for offering 'recharge' sessions when I am depressed and isolating myself from the others; Wing Ng and Chi Pang, for endlessly tolerating just before the assignment deadlines; and Samuel Yu for sharing many experiences. A keen appreciation for friends and acquaintances I have not listed here too.

Finally, my family always immeasurable support and care of me irrespective of my fluctuating health and mood. They have supported in numerous aspects for 21.75 years - take care.

COPYRIGHT INFORMATION

In this dissertation, part of the pedestrian pavements and road data produced by © OpenStreetMap contributors. The data adopt the Open Database Licence (ODbL). Please visit the webpage https://www.openstreetmap.org/copyright for details. The author will upload the derived data with ODbL on the author's GitHub repository (https://github.com/jchan-gi/OSM-Network-Compare).

The source code in this dissertation would make public on the same site. Those materials will use a BSD 3-Clause ("Revised") Licence. Please refer to the uploaded repository after administrative works finished.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASPL ..... Average Shortest Path Length

DD ..... Data Dictionary

DM ..... the iB1000 Digital Map

EPSG ..... European Petroleum Survey Group

GED ..... Graph Edit Distance

GIS ..... Geographic Information System

GISci ..... Geographic Information Science

GIS-T ..... Geographic Information System for Transportation

ISO ..... International Organisation for Standardisation

OGC ..... Open Geospatial Consortium

OSM ..... OpenStreetMap

the site ..... the selected site at East Tsim Sha Tsui

VGI ..... Volunteered Geographic Information

LIST OF COMMON SYMBOLS USED IN GRAPH

A..... Adjacency Matrix

$\mathbf{a_{ij}}$ ..... Adjacency Matrix element at row i and column j

N..... Set of nodes

n ..... number of nodes

d(i,j) ..... Minimum path length between node i and j

1. INTRODUCTION

This section will focus on the background and motivation of conducting this inquiry, the intended objective of this project, and the according structure of this dissertation.

1.1. Background

The Office of the Chief Executive (2015, 2016, 2017) have initiated policies for constructing a Smart City in their Policy Addresses. Those policies lead to the production of Smart City Blueprint for Hong Kong (Innovation and Technology Bureau, 2017). The blueprint listed actions from Six dimensions of smart City proposed and quantified by Cohen (2014). In particular, the blueprint (2017, p. 8) describes few fields to be accomplished, for instances, encourage the distribution of open data, and initiate the "Walk in HK" campaign. One theme in the campaign, "Make it smart", specifies the authorities should consider providing user-friendly information on walking routes.

Interestingly, most routing and navigation application common in Hong Kong, *neither* provide open data about the walking route *nor* provide walking route recommendation. The public may doubt that there is *not* enough public information on the walking path in a useable format. Indeed, even the Government of Hong Kong Special Administrative Region does *not* disclose the available pedestrian walking pavements to date, despite evidence shows that the authority is holding some relevant information already (GovHK, 2018).

One may recommend that VGI could be a database for the current application and research projects. VGI has become a hot topic in the geospatial industry after Goodchild (2008, p. 2) have observed that "the widespread of engagement of a large number of private citizens … in the creation of geographic information". Since then, various authorities, including the authorities responsible for mapping, have developed programs to work with the local communities (Ordnance Survey, n.d.; USGS, n.d.). The ideas of VGI is *revolutionary*, as described by Goodchild (2008). Indeed, the *community contributed* character of VGI may be a match - incidentally or not - for the Smart City 3.0, where Cohen (2015) stressed the importance of citizen co-creation.

Having a more in-depth look in the available dataset, OSM, in Hong Kong was lack of pedestrian topology format from first-hand research. Network model in GISci backed on a mathematical graph at least composed of topology, and connectivity and play an essential role in flow entity model in GIS-T (Fischer, 2006). Flow entity, in the situation of a pedestrian walking path, is the pedestrian. Without a network topology, the aims of evaluating pedestrian routing performance cannot be done. Hence, conversion of the line to the network must be finished in order to build a model available for network analysis successfully. Indeed, some research on Hong Kong, which used OSM data already applied the conversion (Higgins, 2019).

VGI now seems like a solution; however, Goodchild (2008, p. 2) have also emphasised that, "… [the volunteers] are largely untrained, and their actions are almost always voluntary, and the results may or may not be accurate."

The evolution and popularisation of VGI raised along with their quality issues.

The ISO 19157 standards clearly defined the data qualities of spatial data, including VGI. Those qualities, including **internal quality** and **external quality** (usability). The *internal quality* of **positional accuracy**, **thematic accuracy**, and **completeness** have been extensively researched after the article by Goodchild has been well spread (Antoniou & Skopeliti, 2015; Girres & Touya, 2010; Hakley, 2010; Kounadi, 2009; Arsanjani & Vaz, 2015; Antoniou, Corcoran, Mooney & Winstanley, 2010, Boeing, 2017) and generally give positive results. Although a group of researchers have proven the position accuracy and completeness of VGI, few of them like Boeing (2018) worked on the descriptive statistics of topology properties and connectivity of the VGI in the US. Still, the most critical questions, that is, the topology and connectivity quality (defined as "**logical consistency**" in ISO 19157), the most critical measures for network topology, of VGI *versus an official-quality product* have not been answered. For our cases, the suitability of VGI in pedestrian model formation is an *external quality*. However, the result of the application hugely *depends on the topological accuracy* to provide results rely on network analysis, e.g. optimum pathfinding.

Hence, this dissertation aims to develop a methodology to verify whether VGI's dataset is topologically comparable to professional standards. The pedestrian pavement line features in OSM, a platform for VGI, would be converted into topology format equivalent to network model in our investigation. A site at East Tsim Sha Tsui has been selected for verifying the

methods, due to its topological complexity with many different levels. Finally, a conclusion would be drawn whether the internal quality of VGI sufficient for uses, e.g. performing research or providing infrastructure for a smart city.

1.2. Aims and objectives

The dissertation aims to examine the accuracy of VGI through a method to compare the topological quality of network model converted from VGI line features. The objective of the research includes:

1. To design and implement a procedure to compare the topology properties of a network from VGI versus the manually created network;
2. To perform a case study to evaluate the topological accuracy of VGI network, based on the algorithm in point 1, from conversion based on topology analysis and connectivity;
3. To investigate and conclude the capability of those converted data from logical consistency in point 2; and
4. To recommend directions for future investigation.

1.3. Structure of this dissertation

This dissertation is divided into six sections, including this section.

Section 1 (current section) states the background and motivation of conducting this investigation, and aims and objectives of this dissertation. Also, this section describes the structures of this dissertation.

Section 2 summarises several aspects of the dissertation from past literature. Those aspects include the graph theory, application of network in GIS-T, VGI, ISO 19157 and relevant research, finally topology analysis. The perspective of validating the logical consistency would be developed again with a detailed review of current work.

Section 3 explained the methodology and illustrated the implementation. This session is going to describe the study area, hardware and software requirement and data requirement. Then, we will compare the metrics available and based on the characteristics of those algorithms; we would develop our resolution to the issues. The algorithm and final implementation will be depicted in both textual and graphical format.

Section 4 describes and illustrates the result of the topological analysis between two primary sources of data. Afterwards, this dissertation discussed the usability of converted VGI data based on the result.

Section 5 concluded this dissertation. Key findings would be highlighted in this section. Then the project would be evaluated for limitation and further

refinement. Finally, the future outlook on evaluating topological accuracy and external logical inconsistency would be discussed

Section 6 listed the reference used in this dissertation.

Appendix A included the source code of the dissertation. The data and the source code would be available on https://github.com/jchan-gi to conform with the licenses of OpenStreetMap data after all administrative works are done.

2. LITERATURE REVIEW

The network model, such as the road network, is a properly-modelled infrastructure supporting GIS-T application like navigation and spatial interaction research (Shaw & Rodrigue, 2012). Other than transportation, telephone or internet network, even social network, all shares the same mathematical theory behind – the graph theory. With the understanding of the basic principles of graph theory, we could further understand how network model models pedestrians' walking path.

Next, the VGI would be defined. Several vital issues like accuracy problem would be addressed with the ISO 19157 standard. We would have an overview of how previous works deliver lines to network conversion and how it is different from our workflow. Finally, the current research on VGI data quality would be summarised in order to establish our perspective of validating topological accuracy of VGI data.

2.1. Graph theory

It was believed that the *Königsberg Bridges* problem is the origin of modern graph theory. Pregel River cut Königsberg into four parts, while seven bridges interconnect the parts. It was a problem to know whether it would be possible to walk across the seven bridges at once only. Later, Euler, a famous mathematical from Swiss, proved such a walk did not exist. He constructed the base of today's graph theory by the graph he had used. However, the underlying mathematics did not formalise until Martin (1979) annotated a **graph** with the following definition, "A graph G = (N, A) consists of (i) a finite set $N = \{n_1, n_2, \ldots, n_n\}$, whose elements are called **nodes**, and (ii) a

subset A of the Cartesian product A×A, the elements of which are called **arcs**."

After twenty-year of development, a new concept, **network**, had been developed extensively. Carter and Price (2001) defined the most critical refinement, that a *directed graph*, and represent a *system* or *information* as a *network*. In the paper, they also use multiple names of nodes and arcs; for instances, arcs are also known as **edges**; while they noted nodes are interchangeable with **points, junctions**. Moreover, the network adopted in GISci has embedded with a value or cost in the data model (Fischer, 2006). Hence, a network, whether it was specialised for GISci or not, is an overall refinement to the traditional graph suggested by Martin. Figure 2.1 illustrates the differences between a simple graph defined by Martin (1979) and a network defined by Carter and Price (2001), while Table 2.1 summarised the differences in tabular format.

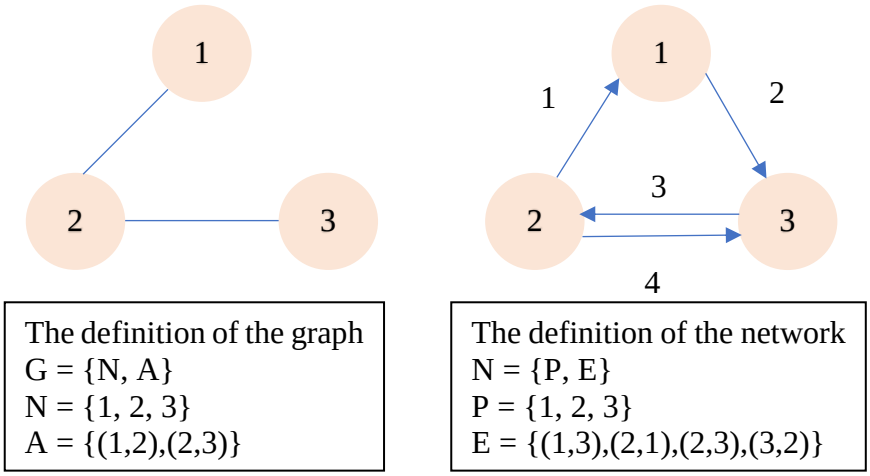


Figure 2.1(a) (Left) – A graph with three nodes and three edges; Figure 2.1(b) (Right) – A network with three nodes and four directed arcs

Differences	Traditional Graph	Network
Directional edges	Implicit	Explicit
Cost / Values	Implicit	Explicit (if needed)
Usage in GIS-T	Less frequent	More frequent
Used in this project	**Default**	Extensible

Table 2.1 – Tabular comparison between graph and network

The graph and network share the same fundamentals. Hence, many properties and modelling techniques for the graph are still useful. Some selected knowledge on data model and network data topology involves in our dissertation will be explained in the following subsection.

2.1.1. Adjacency matrix

Network topology refers to as the inter-relationship between the edges and nodes of a graph. In network analysis, Bell and Iida (1997) summarised in the book that a matrix of a number could define the existence of this relationship. In practice, the graph or the network could have a tailor version of the matrix. The difference leads to two standard versions of simple data modelling for network topology, namely the **adjacency matrix**.

The adjacency matrix, also known as the connection matrix, measures the connectivity of a graph. It is defined as follow: A square matrix A with order equals to the number of nodes n. If i and j are defined as point $\{p_1, p_2, \ldots p_n\}$, then the element of the matrix a_{ij} is equal to 1 if i is connected to j or 0 if they are either not connected or $i=j$ (Chartrand, 1985). Figure 2.2(a) defined the adjacency matrix for the graph in Figure 2.1(a).

Figure 2.2(b) illustrates the incidence matrix of the network in Figure 2.1(b).

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Figure 2.2 – Adjacency matrix of Figure 2.1(a)

2.1.2. Categories for different graph models

Rodrigue (2017) summarised four basic categories of graph model applicable in transport geography, the theory behind GIS-T. The four models described are listed in Table 2.2

Category	**Types**
Planarity	Planar graph
	Non-planar graph
Number of types	Simple graph
	Multigraph

Table 2.2 – Two basic class of graphs

A planar graph is a type of graph such that no graph edge could intersect other edges (Harary, 1994). This definition implicitly introduces a restriction, that a node must exist on any intersections point of edges. This restriction also means that such a graph cannot contain relative height information (Rodrigue, 2017). In contrast, a non-planar graph does not have such a restriction (Harary, 1994). Figure 2.3 demonstrates the two types of graphs.

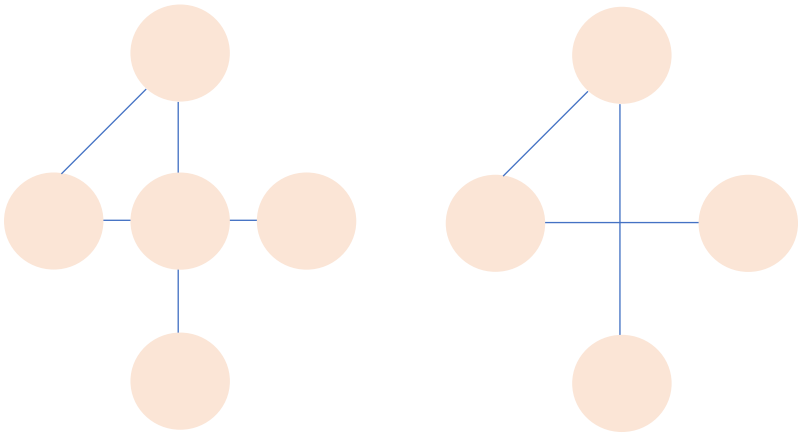


Figure 2.3(a) (Left) – A planar graph.
Figure 2.3(b) (Right) – A non-planar graph.

A multigraph is a class of graph which allows or requires multiple edges between two nodes (Rodrigue, 2017); Conversely, a graph which does not allow multiple edges is a simple graph. Figures 2.4 shows a multigraph. The multigraph proposed two different approaches to the same origin and destination. One example would be a barrier-free slope for wheelchairs versus stairs: they allow users to travel up and down in different ways.

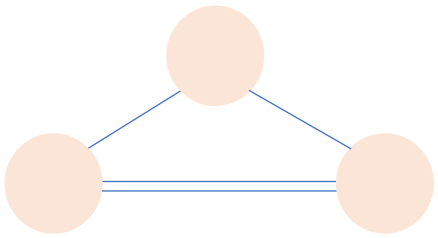


Figure 2.4 – A multigraph.

2.1.3. Metrics on measuring network

Rodrigue (2017) classified various criterions for describing the properties of the network. In addition to Rodrigue's books, some researchers used another set of criterions on evaluating urban road topologies (Jiang & Claramunt, 2004; Jiang, 2007; Porta, Crucitti & Latora, 2006). Moreover, our research will compare two different networks. Therefore the most common metric for comparing network in the past, graph edit distance, should also be included. Table 2.3 has stated the measures in various research, and we would briefly describe those measures below. We will describe, in a logical order, to describe those measures.

Category	**Metric**
Assortativity	Average degree connectivity
	Transitivity (Clustering coefficient)
Centrality	Betweenness centrality
	Closeness centrality
	Degree centrality
	Efficiency centrality
	Straightness centrality
	Information centrality
Hierarchy	Flow Hierarchy
Shortest Path	ASPL
	Efficiency
Similarity	Graph edit distance

Table 2.3 – Metrics for characterising a graph

2.1.3.1. Degree and average degree connectivity

Degree *k* defined as the number of edges connected to node *m*. Hence, the **average degree** *k* of the whole graph *G* could be expressed as

$$\bar{k}(G) = \frac{1}{n}\sum_{i}^{n} k_i \qquad (2.1)$$

In Equation (2.1), a_{ij} is the element in adjacency matrix A defined in Section 2.1.1. However, this measure the current node only. The **average degree connectivity**, also known as **average nearest neighbour degree**, measures the mean degree of the neighbours.

$$k_{nn,i}(G) = \frac{1}{n}\sum_{j \in a_{ij}=1} k_j \qquad (2.1)$$

2.1.3.2. Shortest path, diameter and ASPL

The **shortest path,** denoted by ***d(i,j)***, is a particular case of the optimum path, which is defined as the path distances traversed between two nodes is the shortest. In graph topology, it is defined by the number of node/edges being traversed. It could use the geographic distance, time or other costs in flow entity model applied by GIS-T analysis; hence more broadly an optimum path (Tang, 2005). However, they are derived from the basic definition from topology and therefore not a significant concern in this dissertation.

Similarly, **ASPL** *p* of the whole graph is defined by a formula

$$p(G) = \frac{1}{n(n-1)} \sum_{i}^{n} \sum_{j,j \neq i}^{n} d(i,j) \qquad (2.3a)$$

In converse, **efficiency** *E* is defined as the inverse of distance, i.e.

$$E(G) = \frac{1}{n(n-1)} \sum_{i}^{n} \sum_{j,j \neq i}^{n} \frac{1}{d(i,j)} \qquad (2.3b)$$

In short, one is a maximum measure, while one is an average measure.

2.1.3.3. Clustering coefficient

Clustering coefficient C measures how the graph clustered by calculating the "probability that two neighbours of a given node are linked together, that is, a ratio of the number of actuals edge to the possible edges" (Jiang, 2007, p.651). In that article, the average clustering coefficient for all nodes is being used:

$$C(G) = \frac{1}{n} \sum_{i}^{n} \frac{\#\ of\ actual\ edges}{\#\ of\ possible\ edges} \qquad (2.4)$$

2.1.3.4. Centrality

The centrality measures how the node located in centre topologically. Central nodes in a network are more influential than the others in several aspects (Bavelas, 1948). Freeman (as cited in

Porta et al., 2006) integrated previous researches on the properties and defined the first three *centrality* indices. The three centrality indices are **degree centrality**, **closeness centrality**, and **betweenness centrality**. The three centralities have been summarised by Porta et al. (2006, p.709) as follows:

$$C_i^D = \frac{\sum_j^n a_{ij}}{n-1} \tag{2.5}$$

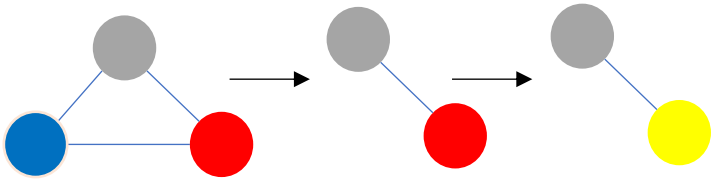
$$C_i^C = \frac{n-1}{\sum_{j,j\neq i}^n d(i,j)} \tag{2.6}$$

$$C_i^B = \frac{1}{(n-1)(n-2)} \sum_{j,k;j\neq k;j,k\neq i}^{n} \frac{n_{jk}(i)}{n_{jk}} \tag{2.7}$$

In Equation (2.7), the n_{jk} is the number of the shortest paths between point *j* and *k*, while $n_{jk}(i)$ contains the number of the paths pass through *i* only.

At a later epoch, researchers have resolved the problem on the disconnected graph by inverse the distance in path length (Latora and Marchiori as cited in Porta et al., 2006). Those centralities are *efficiency centrality*, *straightness centrality* and combination of two, *information centrality*. Those centralities would require Euclidean distance in geographic space hence out of our scope.

2.1.3.5. GED



G_1 G_2

Figure 2.5 – Two graphs with GED of 2.

The measures above apply to one graph only. There is a novel algorithm on comparing two graphs, namely **Graph Edit Distance**. The algorithm tries to add, remove or substitute nodes or edges to match another graph. Figure 2.5 demonstrates the algorithm of GED. GED is computation-intensive since it searches for all possible actions. Hence, recent researches have shifted the focus on less expensive algorithms, for example, Spectral Graph Theory.

2.1.3.6. Flow hierarchy

A directed graph could have cycles. The percentage of edges not in the cycles would be the **flow hierarchy** of the graph (Luo & Magee, 2010). The metric defined the percentage of weakly connected edges.

2.1.4. Our perspective based on reviewing graph theory

In the previous subsections, we have reviewed the principle of graph and networks.

Definition of traditional graph defines sufficient graph topology for our scenario. Tradition definition of a graph against a network differs by directed graph and use scenario. A network is a refinement to a graph and most applicable in real-world application space. Since this dissertation focused on the topological accuracy of VGI pedestrian data – which do not have direction, hence a traditional graph already fit the purpose.

There are various classes of graphs. Intersections of edges could be allowed or forbidden; also multiple edges for the same origin and destination could be configured. In the site, there are various footbridges and tunnels. Hence a non-planar graph is unavoidable. Part of the site constructed accessible slope for the need, hence caution have to be raised in site visit stage.

Finally, various measures are available for characterising a graph in graph level and node level. At this stage, we would delay our selection of benchmark to later epoch, to be specific, after we have reviewed the current research on VGI accuracy. However, a straightforward benchmark is preferred for more understanding of the research problem (Section 3.1).

After all, the graph theory has been extensively applied in various domain, for instance, internet and social network. Graph theory also plays a significant role in GIS-T. Our application involves network analysis of transportation system. Hence a brief overview of GIS-T would be appropriate in the next subsection.

2.2. GISci, GIS and its application in GIS-T

This dissertation has mentioned a legion count of GISci, GIS, and GIS-T, but still has not tried to address these terms. Also, how they involve in the study is unexplained. In this subsection, we would describe and explain those interrelations with existing literature.

In the preface of a book from Longley, Goodchild, Maguire, and Rhind (2005, p. xi) believe that GIS is a paradigm of "relating scientific principles" to solve "science" problem. The GIS-T is the corollary of an information system solving scientific problems in Transportation Geography. In the book from Millar and Shaw (2001), they have considered explaining the science of transportation underneath the GIS-T, which matches the trend of establishing science nature with GIS. Those introductory chapters of GIS books have clearly defined the inseparable lineaments of a science subject. In the following subsections, the principles and application of GIS relevant to our study will be summarised based on hierarchical order.

2.2.1. GISci, GIS and their network model

GIS has no unified definition. Maguire (1991) defined GIS as an information system based on and valued on the spatial element of data. GIS users may require numerical analysis on the geographic information and finally present those data in a cartographic format (the GISci). Opensaw (1991) defined those analyses as spatial analysis. Among those analyses, "Network analysis and graph-theoretic methods" have been ranked first in methods for line data analysis by him. It could demonstrate that

the network analysis functions, first appeared on Lupien, Moreland and Dangermond (1987), have been highly anticipated already.

There were numerous applications of network analysis described by Lupien et al. (1987) already. However, as Longley et al. (2005) explained by example, what matters the network analysis example is their structural properties (connectivity), or in a border term, topology. Curtin (2007) had also briefly stated those descriptions available on graph theory, the fundament of the GIS network. Hence, one primary focus on the geographic line features would be their structural properties, also known as topological analysis (Jiang & Claramunt, 2004; Jiang, 2007). In mathematicians' perspective, numerous of those description techniques were already well defined. However, in planners' and geographers' perspective, past textbook and researches have suggested some simple metrics in Section 2.1.3 (Boeing, 2018; Jiang & Claramunt, 2004; Jiang, 2007; Rodrigue, 2017). Hence, our dissertation would select metrics based on them.

In contrast to structural properties, the flow model is also essential in network model defined by Lupien et al. (1987). In short, any entities that utilise those network would be their flow. In the early stage of network analysis, the research focused on the maximum flow of the network, which the maximum flow is essentially minimum cut or minimum cost (Ford, Jr. & Fulkerson, 1962). The cost could be the impedance of each edge, turn at

nodes, and more. Hence, the flow model relies on the structural configuration of the network. Therefore, this dissertation *did not cover* the *flow model* yet.

2.2.2. GIS-T specialisation on the network models

In contrast to the diverse GISci topics in GIS, GIS-T focused the discussion area in transportation. Millaw and Shaw (2001) have summarised the GIS-T model as three-component: a datum supporting all design and implementation; a network for the connection and a linear referencing system for locating point or section in the network. Figure 2.6 illustrates the relation of them.

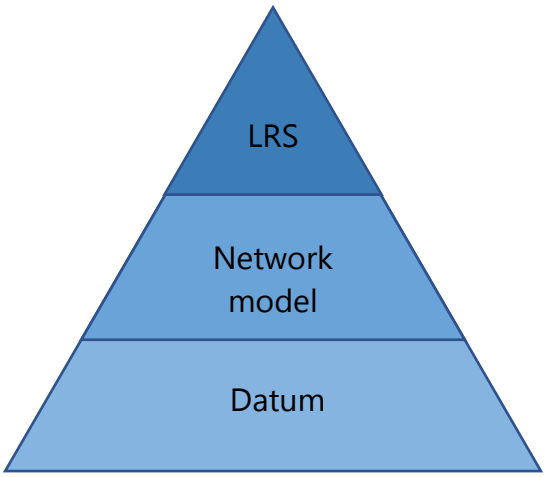


Figure 2.6 – GIS-T data model

In traditional GIS-T design, academia would use the simple node-arc network to implement GIS-T data models. There are various issues on the GIS-T network model. The most critical

issue discussed would be the representation of lanes and intersections.

Consider a road similar to Figure 2.7; the digital representation of the road could be a single lane approach (left) or a multiple-lane approach (right) in GIS-T network model. This modelling decision is intrinsically a problem of allowing bidirectionally or aggregation of line features.

For multiple intersecting node problems, one could model the intersection of the road in multiple ways. One great example would be in Figure 2.8, where the intersection of two dual-lane carriageway could have multiple modelling outcome.

Figure 2.7 – Single lane representation (left, red) and multi-lane representation (right, cyan)

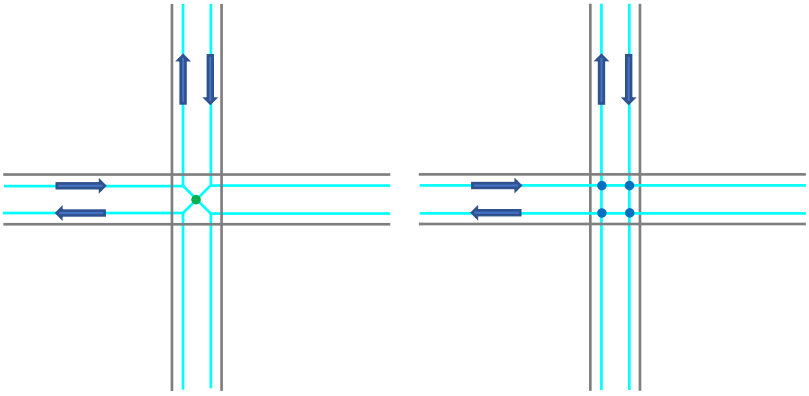


Figure 2.8 – Single node intersection (left, green node) and multi-node intersection (right, blue nodes)

Those representation problems are sourced from how to model reality into a data model. There is no definite winner for these problems; however, a consistent model is always preferred in such a scenario.

One would implement those data models into physical hardware and software after all modelling is done. In GIS, a relational database is the first choice for those data structures. As Millar and Shaw (2001) explained that, those networks have an arc, node and a dedicated **turntable** implemented in a relational database. The rationale of the turntable would be the turn restriction and cost for real road. It is noted that turn restriction maybe not applicable in most pedestrian pavements.

Hong Kong is a too small place to drive for all the people, hence walking is essential for reaching public transport stop. However,

that book did not cover much about pedestrian model. Instead of a network model, originally a raster was being used to model off-network movement like walking (Upchurch, 2004). One may be expected that pedestrian movement is unpredictable and not constrained to the network only. However, in a later epoch, in a peer-exchange report for the U.S. Department of Transportation, the authorities have inventoried the pedestrian and bicycle as a network model for GIS analysis (U.S. Department of Transport, 2009).

Hence, in an annal written by Shaw (2010), he recommends a fused vector and raster approach for modelling off-network movement like pedestrian movement. This lead to the contribution of the pedestrian network in GIS-T is inevitable. Nevertheless, the problem definition in this dissertation discovered that a suitable authoritative network may or may not be available. This lead to academia utilises citizens co-created information available in both vehicular roads or pedestrian pavements, namely, VGI.

2.2.3. Our perspectives based on reviewing the literature

Those works of literature summarised the application of flow-entity network model in GIS, GISci, and GIS-T. From the summary given, this dissertation believes that:-

a) Network structure and flow model is essential in GIS network model; however, the flow model depends on the network structure;
b) The network model is critical in modelling pedestrian walking paths;
c) Consistent modelling techniques are required; and
d) Modelling the walking path by a network model are less studied. The reason may be that the foreign country is too big to walk.

2.3. VGI and OSM

The renowned paper from Goodchild (2008, p.2) observed that "the widespread of engagement of a large number of private citizens … in the creation of geographic information" and describe this phenomenon as "revolutionary". The public co-creation product has named as VGI. Indeed, the authorities, such as the United States Geological Survey and Ordnance Survey, have also acquired data sourced from VGI. VGI has become a vital data source from both public, academia and authorities.

VGI is only the concept or product of the sharing and co-creating geospatial information. However, platforms are required to accomplish those actions. There are few platforms for the job, and the most visited VGI platform is OSM.

In order to understand the principle of how we convert the data, it is better to understand the underneath storing mechanism of OSM. OSM has developed its standards for storing spatial information. From the explanation on

Mapbox (n.d.), the three major categories are **nodes**, **ways**, and **relations**. Nodes are all point objects; Ways are all linear objects and polygon boundary; while relations define the structured relation between multiple objects. One could utilise those data structures to reconstruct a database for point, line, polygon and network features.

There are two major ways to reconstruct those data structures into the intended format, in our case, network format. The first way is a direct query to the overpass API as defined by the online documents. Olbricht (2015) explained that this method could obtain source data directly. However, those source data are often unorganised and not standardised. From this dissertation trials, the data are frequently missing even querying conform to the data dictionary. The second way is applying existing tools to construct the data, while the most renowned tool for our study is OSMnx. Boeing (2017) have already developed OSMnx, a handy tool for extracting relevant network features from OSM. The tools have extracted and reconstructed the network structure from the above objects. The process is all automatic and gives acceptable completeness result compared with the Overpass API.

Per previous sections, one could understand that network structure is equally important with the flow model in the GIS network. Hence, the network structure requires high accuracy to portraying actual ground truth. One may raise a question like "what classes of accuracy are available to the network dataset or even the VGI data?" Uncertainty is an intrinsic element to spatial data in early studies (Maguire et al., 1991). In the following subsection, we would have a view of the types of accuracy available.

2.4. Accuracy standards and ISO 19157

Uncertainties have been well discussed by an edited chapter from Chrisman (1991). He has summarised four primary components of uncertainties. At a later epoch, the ISO have published ISO 19113, 19114 and ISO/TS 19138 for various data quality measurement scheme. Those standards have been superseded by ISO 19157, namely Geographic Information – Data Quality. In ISO 19157, the four basic categories have been slightly modified and extended into five component for a full assessment on usability. In this section, those standards would be briefly described and linked those measures into research on VGI data quality in the last section on literature review.

2.4.1. Original error components from Chrisman (1991)

Chrisman (1991) have analysed the error components of standard geographic information and listed the following four error component:

a) Positional
b) Attribute
c) Logical consistency
d) Completeness

Positional accuracy is a very common accuracy related to the actual location versus the mapped coordinates. However, Chrisman also stated that GIS does not portray the actual location without an error. This dissertation might face a similar issue, like rounding error problem.

The **attribute accuracy** is an error composed of two parts. The first part is the conversion of metrics: tools or techniques may have a presumption, failing to meet the presumption will lead to invalid results and attributes. The second part would be categories error: the categories may be wrongly classified or attributed for a geospatial data. Both problems lead to low attribute accuracy.

Logical consistency would test the implicit rules on spatial information. Chrisman (1991) explained that the topological model or consistency test is often a tool to test positional or attributes. Chrisman (1991) used an example of "buoys cannot on dry land" as an example test case for logical consistency. Another example would be two sliver lines. This dissertation has covered a large part with the graph theory. The graph structures is a topological property hence classified as logical consistency.

Completeness is a measure of coverage. Chrisman (1991) listed an area of non-coverage or too small for display would produce incompleteness problem.

2.4.2. ISO 19157

ISO 19157 reused and modified the US standards and listed five significant internal elements in spatial information quality. Those internal quality defined by ISO 19157 are:-

a) Positional

b) Thematic

c) Temporal

d) Logical

e) Completeness

The significant changes would be broadening the attribute accuracy to thematic and adding a time quality for the trend of spatial-temporal analysis. Finally, ISO 19157 have linked those internal qualities to usability related to each use case (or external quality). Figure 2.7 demonstrates the definition.

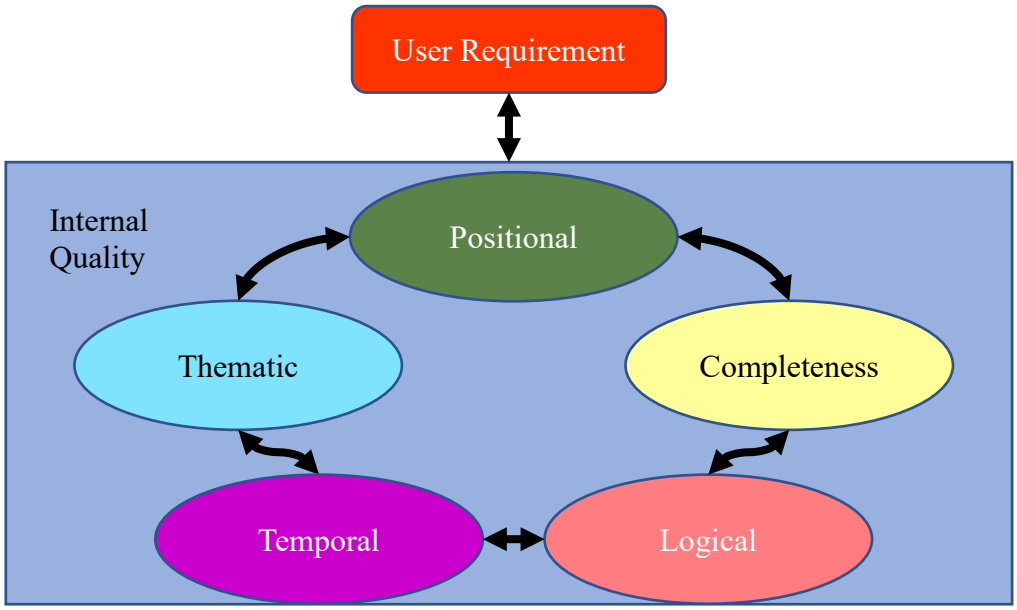


Figure 2.9 – ISO 19157 standards definition.

2.5. Relevant research on VGI data quality using ISO 19157 standards

In an edited chapter by Fonte et al. (2017), they have an excellent understanding of the VGI data quality definition. The previous section we have

simplified and extracted the origin of those definitions. They have also listed a table (Table 2.4) for requirements for that assessment:-

ISO quality elements		Requirements
Internal quality	Positional accuracy	• Data specification • Existence of reference data with similar characteristics and valid time frame
	Thematic accuracy	
	Completeness	
	Temporal quality	
	Logical consistency	• Other data of the same source or independent data
External quality	Usability	• Specification of user needs

Table 2.4 – ISO quality elements and their requirements

Apart from Fonte et al. (2017), Antoniou and Skopeliti (2015) have also summarised some measures and indicators of VGI data quality using ISO 19157 standards in past studies. In addition to those summaries, there are few more investigations of VGI data quality in the past.

2.5.1. Studies on each component of accuracy

First, for **positional accuracy**, there are piles of papers analysed the properties already. The two most crucial pieces of research are Hakley (2010) and Girres and Touya (2010), which generally give positive results on positional accuracy.

Second, for **thematic accuracy**, there is extensive research on the topics, such as Kounadi (2009), Girres and Touya (2010), and Arsanjani and Vaz (2015). Other researches have also proposed some methodology to identify and pinpoint the discrepancies.

Third, for **completeness**, since complete data is the base to provide sufficient enough information, the study of completeness has been started very early stage. Relevant researches give useful results, such as Kounadi (2009), Hakley (2010) and Girres and Touya (2010).

Fourth, for **temporal quality**, despite spatial-temporal analysis and modelling becomes the trend of recent studies, few studies have focused on the temporal quality of VGI. Despite some critical benchmarks and findings from Girres and Touya (2010) and Antoniou et al. (2010), there is not much breakthrough recently.

Lastly, for **logical consistency**, this dissertation proposed two types of logical consistency from Table 2.4. One could check the topology or conduct other studies *based on the same source* – this is an internal check hence **internal logical consistency**. In contrast, one could check the VGI against with the external sources, which is **external logical consistency**. This dissertation clarified the differences between two due to the current investigations focused on internal logical consistency only. For example, Corcoran et al. (2010) have studies the internal topological consistency of segments sharing a boundary. The comprehensive benchmarking by Girres and Touya (2010) have studied the theme/tag/schema consistency. Even Boeing (2017), who create OSMnx introduced in Section 2.3, have focused on the internal logical consistency only. Whether the VGI portray the ground

truth, that is, a reference data is unclear from the past investigation. Interestingly, there are already recent researches on VGI data using the OSMnx despite the uncertainty in external logical consistency, such as Higgins (2018).

Finally, for **usability**, it is indeed defined as overall assessment with all internal accuracy, finally linked the metric with a user requirement analysis. The study of Girres and Touya (2010) are the most impacted research on usability; however, still not considering external logical consistency.

2.6. Problem definition of this dissertation based on the literature

From the elaboration in the last few sections, we understand that the network model is a significant role of GIS-T data model since the datum is usually well-defined and maintained by surveying authorities. Although the raster model was being used in modelling off-network paths like walking, recent studies and research also prove the network model is irreplaceable in modelling some off-network movement like walking. However, the network model is sometimes unavailable for several reasons. One might consider using VGI data set to replace the official data.

VGI is community-contributed data, and those data might result in poor standards. Through verification must be done before applying the VGI, although many studies have been conducted for VGI data quality based on ISO 19157 scheme definition. Most elements are well studied and give a positive result on applying VGI. However, this dissertation also has discovered there

are very few assessments on *external logical consistency* with VGI, especially for network model. Despite internal validation of logical consistency means useable data; however, how well the network model portrays the ground truth is unclear.

There are a few critical problems in the research question. The first would be should a new metric to be created for this question; the second would be should we adjusted our analysis result for other quality assessment criteria in ISO 19157.

From the past literature on researching VGI data quality, academia usually creates a new metric or reuse existing metric for the analysis (Antoniou & Skopeliti, 2015; Girres & Touya, 2010). The non-standardised and inexpert nature of VGI requires diversified analysis techniques and methodologies. In order to deal with those uncertainties; a new metric may be more accurate to embed the amateur nature of VGI.

The same problem may arise when analysing network structure, a metric of external logical consistency. However, the graph-theoretic methods have supported the network topology and structure much. Although VGI may exist its characteristics in data structures, a new metric is inappropriate if there are sufficient well-known and well-tested metric available unless disproved. Hence, in our analysis, we would reuse existing graph-theoretic method, from Table 2.3, as a trial analyses on external logical consistency.

Moreover, network structure accuracy cannot be simplified as a number. Network structure could be very complicated. Hence, a difference in number

may not indicate incapability of any one of the networks. To be specific, the centralities analysis would show more than a number, hence portray a better "usability similarity" of two networks. Hence, a selected list of analysis instead of a new metric is being created to allow tolerances.

In addition to the metric(s) being selected, another question one may raise on the studies may relate to the completeness of the data set. One may ask that the level of completeness may affect network structures and topology. In a complete analysis of the usability of a dataset by ISO 19157 standards, completeness is already a metric to be investigated. On the other hand, in our study, our objective has focused on the logical consistency but not the completeness, in specific, the network structure accuracy of the VGI data. If our analysis has penalised the completeness in logical consistency, it will become a double penalise in comprehensive benchmarking on usability similar to Girres and Touya (2010). Even Girres and Touya (2010) as well as Corcoran et al. (2010) in studying logical consistency, or Hakley (2010) compares with other datasets, have no correction on completeness. Hence for this investigation, the methodology will not adjust the result using any completeness measure.

To sum up the literature review, this dissertation realised the problems and tried to design and implement a multi-criteria based computation to compare VGI network versus a reference network. Through the metrics and results, the similarity and capability of VGI network could be evaluated by a high-quality reference network. The methods have been implemented as a proof-of-concept using pavements, and the result would be discussed. From the

discussion, we could have a better view of how VGI perform in the site on the network structure accuracy, an external logical consistency measure.

3. METHODOLOGY AND SYSTEM IMPLEMENTATION

This session is going to describe the study area, hardware and software requirement. Then, we will discuss the source data required and how we obtain those data. Afterwards, we will compare the characteristics of different metrics, as described in Table 2.3. Based on the characteristics of those algorithms, we would develop our resolution procedure to the issues in finally.

3.1. Study area

The site of this project is the East Tsim Sha Tsui area. A cyan polygon is indicating the study area in Figure 3.1. Few major vehicular roads bound the site, namely Salisbury road, Hong Chong Road, Hong Tat Path, Cheong Wan Road and Chatham Road South. The site is popular among the tourists and local commuting residents. The site exhibits moderate topology complexity with some footbridges and subways which test the data excellently.

However, the private or indoor area in the site was excluded. The reason is that most of these areas are far complicated than the open, public area. Moreover, those areas are not open for 24 hours. Also, the local VGI community discourages the mapping of private indoor facilities. In summary, three major categories of the excluded area are:

- Bus terminus (indoor, e.g. Mody Road Bus Terminus)
- Private footbridges connecting two buildings (private and indoor, e.g. footbridges between Tsim Sha Tsui Centre and Empire Centre)

- MTR Subways (private and indoor, e.g. East Tsim Sha Tsui station entrance P1 outside Wing On Centre)

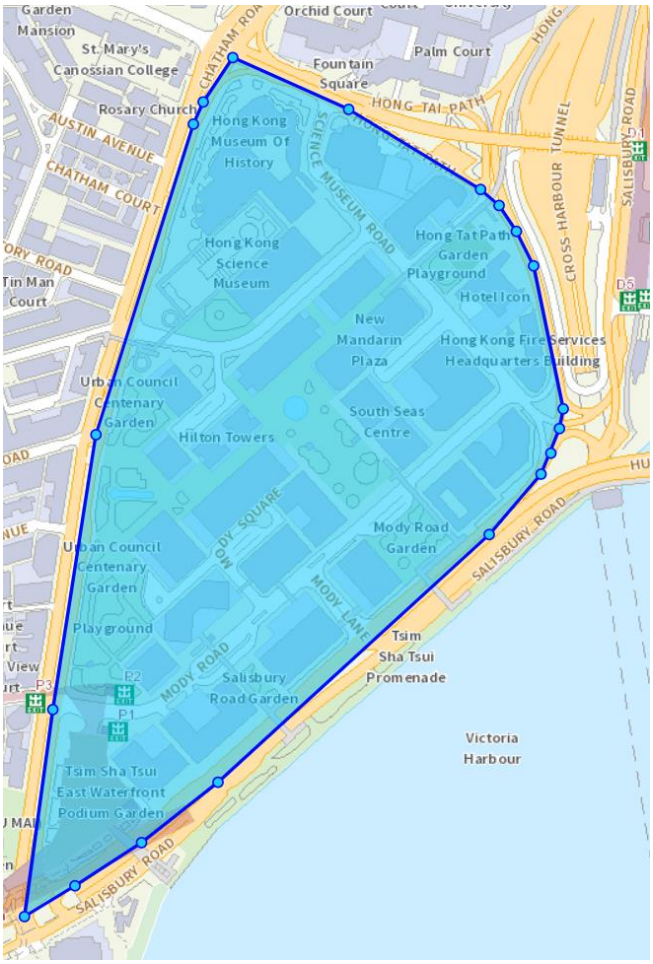


Figure 3.1 – A cyan polygon bounds the site.

3.2. Hardware and Software requirements

An internet-connected computer which could execute Python 3, Jupyter Notebook and able to install packages from Python pip is required for performing analysis. The software outcome of this project, i.e. a Jupyter notebook for executing the analysis, has been developed on Windows 10 version 1903 and tested successfully on Mac OS X 10.14.5.

This software does not intend to bound with specific proprietary commercial software. The user shall install a GIS software for creating the shapefiles, a Python 3 environment and a Jupyter Notebook server. The Python environment shall install the packages of the ***latest version*** listed in Table 3.1. The most critical package for analysis is **NetworkX**. This dissertation has tested our development on ArcGIS 10.6.1, and the Anaconda 2019.03 contain both Python 3.6.8 and Jupyter notebook. The QGIS could replace the expensive ArcGIS for low-cost development and test environment.

collections	fiona	geopandas
itertools	matplotlib	networkx
numpy	osgeo	osmnx
scipy	shapely	Statsmodels

Table 3.1 – Required packages for Python 3

3.3. Data Requirement

There are three two primary sources of the network data to perform the analysis correctly. The first source would be from OSM, a primary source of VGI data; during the second source the from a reference network.

For VGI data required, a polygon enclosing the study area is required to download data from OSMnx. From our discussion in Section 2.3, a simple way (OSMnx) and a sophisticated way (manual reconstruct) could both create a network format for analysis. In this dissertation, the first way, i.e. OSMnx, is being adopted for its simplicity and robustness suggested by Boeing (2017). Through a The network format from OSMnx shall also minimise

other errors, e.g. internal logical inconsistencies already. However, a polygon is required to bound the area to download OSM data. In order to simplify the process behind, the user is required to submit a polygon shapefile to download the OSM data inside the polygon when executing the software.

On the other hand, a line shapefile and a point shapefile are required to establish a non-planar reference network. The user shall prepare a line shapefile denotate the edges, such as pavement and road; and a point shapefile denotate the nodes. A separate nodes file is the critical source to retain the needed intersection only. Without a separate file for nodes will lead to a planar graph, which violates our perspective in Section 2.1.4. There are no specific projection restriction on the reference network format except the projection must be listed on "EPSG Geodetic Parameter Dataset".

Although the data modelling is simple in this dissertation, the modelling process will be listed below for the reader's reference.

3.3.1. System interoperability and data integration techniques

The critical Python package we used, the NetworkX, supports reading the shapefile format, proprietary ownership but open standard. Hence, in the data definition stage, we selected shapefile as a principle data format for the data storage. This decision is aimed to enhance the system interoperability and data integration by using a compatible single format for *direct data translation approach*.

3.3.2. Conceptual data modelling

The conceptual model of the polygon for retrieving the VGI data is simply a specialisation of the polygon defined in OGC in Figure 3.2.

The conceptual model of the point and line in the reference network is composed of multipoint and multiline in OGC standards in Figure 3.3 and Figure 3.4, respectively.

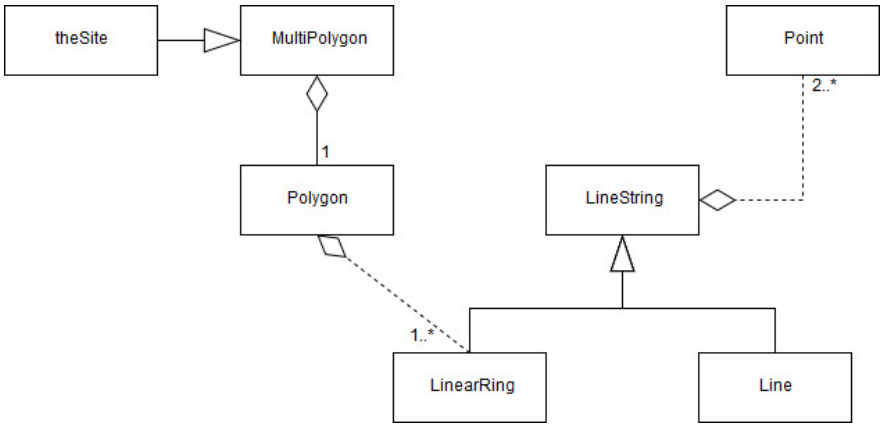


*Figure 3.2 – Conceptual design of the polygon to download OSM data (**theSite**).*

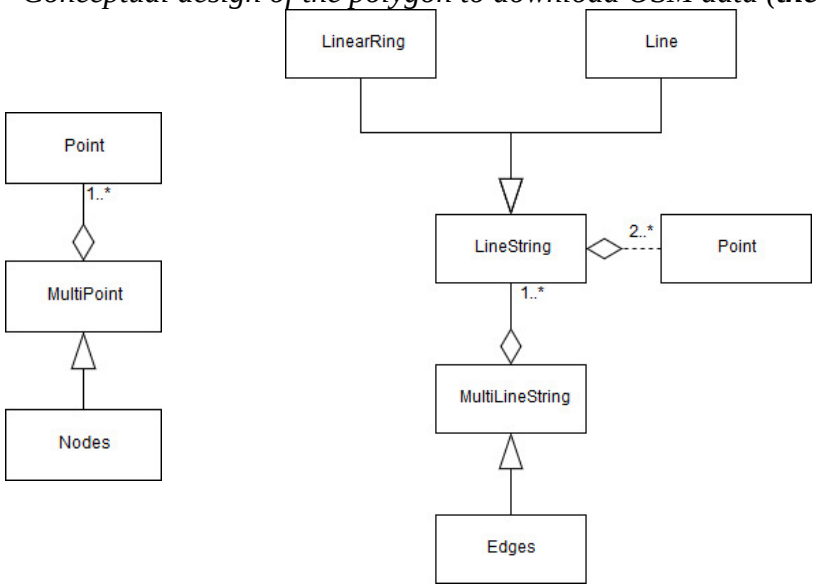


*Figure 3.3 (Left) – Conceptual design of the nodes in the reference network (**Nodes**).*
*Figure 3.4 (Right) – Conceptual design of the edges in the reference network (**Edges**).*

3.3.3. Logical data modelling

The logical design for the **theSite** and **Nodes** will retain their defaults implementation in the logical data model defined by the shapefile standards since no significant need for changes. For **Edges**, we added a ***Z*** for human interpretation for the stage of creating intersections. Figure 3.5 shows the modification. The ***Z*** is a field that used named integer. The integer has specific meaning for their values. The dictionary of the edges is listed in Table 3.2. However, the ***Z*** is intended for establishing network structure and involves in no algorithmic processing in analysis script.

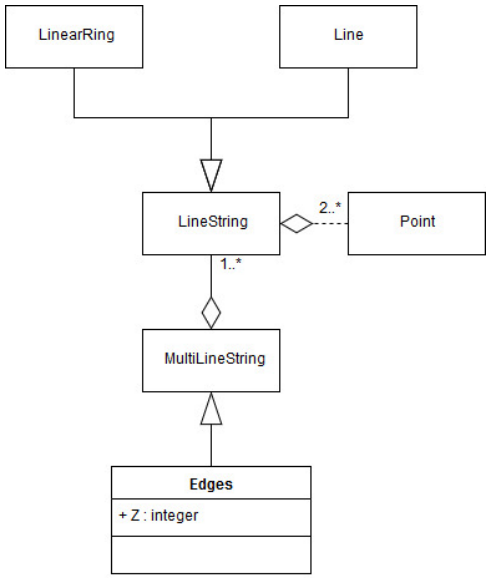


*Figure 3.5 – Conceptual design of the edges in the reference network (**Edges**).*

Attribute	Value	Meaning
Z	-2	Subway
	-1	Stairs/elevator connecting subway and ground
	0	Ground
	1	Stairs/elevator connecting ground and elevated ground/bridges
	2	Elevated ground/bridges

*Table 3.2 – Data dictionary in the logical design of the **Edges***

3.4. Review of available algorithms for characterising a network

After all data definition and modelling task have been done, it is the best time to decide which graph-theoretic algorithm to be applied in evaluating topological accuracy. This dissertation has summarised critical and well-known algorithms, covering nodes, edges and graph level, in Section 2.1.3. Table 3.3 has recapped the information in Table 2.3.

Category	Metric
Assortativity	Average degree connectivity
	Transitivity (Clustering coefficient)
Centrality	Betweenness centrality
	Closeness centrality
	Degree centrality
	Efficiency centrality
	Straightness centrality
	Information centrality
Hierarchy	Flow Hierarchy
Shortest Path	ASPL
	Efficiency
Similarity	Graph edit distance

Table 3.3 – Metrics for characterising a graph

This dissertation would be intended to use *as many algorithms as possible* to differentiate the VGI network and the reference network in different aspect; however, there are implementation constraints with some algorithms listed below.

For **GED**, the literature review has already summarised that the computation time of GED is too expensive. Although GED is the best metric to numerical the different between two networks, it is not practical to wait for the results

with days or even weeks. Hence, it is eliminated from the list for a practical reason.

For **efficiency centrality**, **straightness centrality** and **information centrality**, as they require geographic information and the implementation of geographic coordinate is not guaranteed by NetworkX, the critical package in the analysis. Hence, those centralities would be omitted.

Hence, this dissertation will apply the algorithm listed in Table 3.4 to analysis the accuracy of the VGI network.

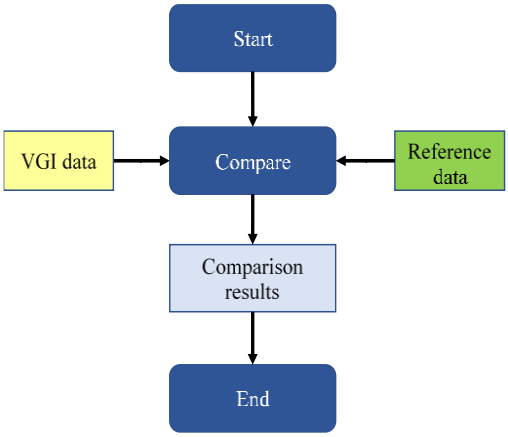
Category	Metric
Assortativity	Average degree connectivity
	Transitivity (Clustering coefficient)
Centrality	Betweenness centrality
	Closeness centrality
	Degree centrality
Hierarchy	Flow Hierarchy
Shortest Path	ASPL
	Efficiency

Table 3.4 – Selected metrics for the evaluation

3.5. System Implementation

After we have discussed all the atomics of preparation work, we would describe the procedure to implement this dissertation's aims and objectives. The conceptual design of the procedure is illustrated in Figure 3.6. Since this dissertation has decided to investigate pedestrian pavements while there

is no official data, this dissertation shall also digitise the pavements as illustrated in Figure 3.7. In the following subsections, this dissertation would describe how to execute the procedure stated in Figure 3.7.



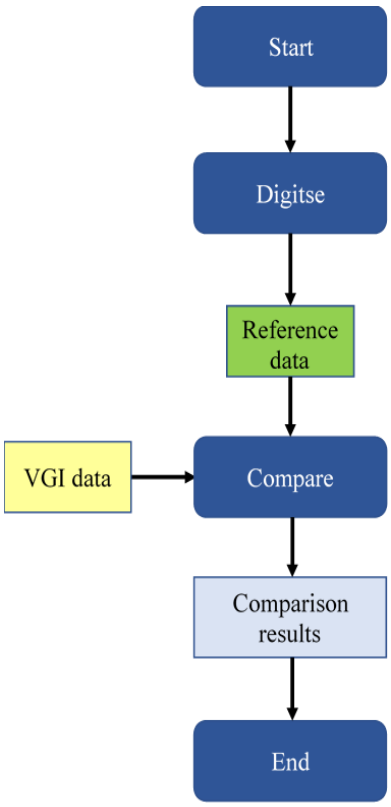


Figure 3.6 (Left) – Conceptual flow of the general analysis process.
Figure 3.7 (Right) – Conceptual flow of the analysis process of this dissertation.

3.5.1. Digitisation

There is two primary digitisation outcome required, namely **theSite** polygon and the **Edges**. First, this dissertation has finished digitising the site boundary (Figure 3.1), and the polygon has already saved to the working folder of the analysis script.

Figure 3.8 demonstrates the digitisation procedures in this dissertation. The outcome of the pavement digitisation will be saved

as a shapefile classed **Edges,** and hence, the pavement is defined using the data dictionary of **Edges** in Figure 3.5.

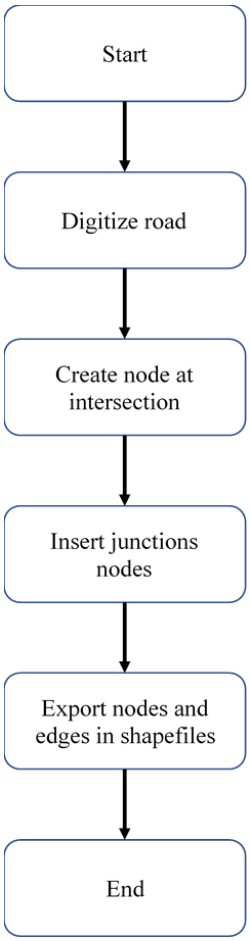


*Figure 3.8 – Flow of creating reference network by the digitisation (the block **digitise** in Figure 3.7)*

3.5.1.1. Scope of digitisation

All the walkable pavements for pedestrian and wheelchairs, as well as public open space, conform with the scope of the project, have been digitised. That is, the following walking paths and pavement will *not* be digitised:

- Bus terminus
- Private footbridges

- MTR subways

3.5.1.2. Method of digitisation

There are two major categories of the walking path on the site. The first class is the roadside pavements; the second class is an open space for free walking. Hence, our digitisation has observed the typical walking trajectories for open space with no specific pavements first. For all open spaces, each location would observe twice per session. The observation sessions are listed below:

- Afternoon, 15 minutes within 15:00 – 17:00
- Night, 15 minutes within 19:15 – 21:00

Then, the walkable pavements are digitised with the aid of iB1000 dataset from the Lands Department and field observation. This dissertation has followed the principle of "major to minor" in digitisation order. Hence, the road is digitised in the following order with the roughly-estimated pedestrian flow with tie-break:

1. Major roadside pavements
2. Streetside pavements
3. Pedestrian only paths
4. Footbridges

Finally, the ***Z*** field will be inserted according to the data dictionary listed in Table 3.2.

3.5.1.3. Rules for digitisation geometry

Literature review on GIS-T in Section 2.2.2 has demonstrated that the geometry of the network has a significant impact on the network model. Hence a digitisation rule must be established in order to standardise the digitisation in this study. Apart from the "major then minor" principle, Three major rules are handling the intersection of the paths:

- Shorter path intersects perpendicularly to longer paths if two are parallel (Figure 3.9a)
- Shorter path intersects perpendicularly to longer paths if two are almost perpendicular (Figure 3.9b)
- Shorter path intersects without changing the angle if the longer paths will intersect shorter paths in angle (Figure 3.9c)

Figure 3.9(a), Figure 3.9(b) and Figure 3.9(c) (Left to Right) – Standard geometry when digitising the pavements

3.5.2. Creation of intersections

After all the edges have been inventoried successfully, the *nodes shall be created now*. The procedure is to create a point at each

intersection. Afterwards, delete incorrect intersections manually. Incorrect intersections are the intersection should not exist, such as the intersection between an overhanging bridge and the ground. During the process, a ***Z*** value is critical to perform database query in order to find non-existing intersections.

This dissertation uses ArcGIS 10.6.1 software to digitise the edges. The fastest way to generate a point at the intersection would be generating a "Geometric Network". Leave default setting when creating a geometric network should already produce a topologically correct network and intersections. However, this dissertation still performs the query to ensure all intersections are correct. The topology is correct if the junction exists on the ***Z*** value of the **Edges** only differ by 0 or 1. After all intersections (Junction) have been verified, we could *export* the edges and nodes to the working folder for the analysis script. Figure 3.10(a) demonstrates the completed network.

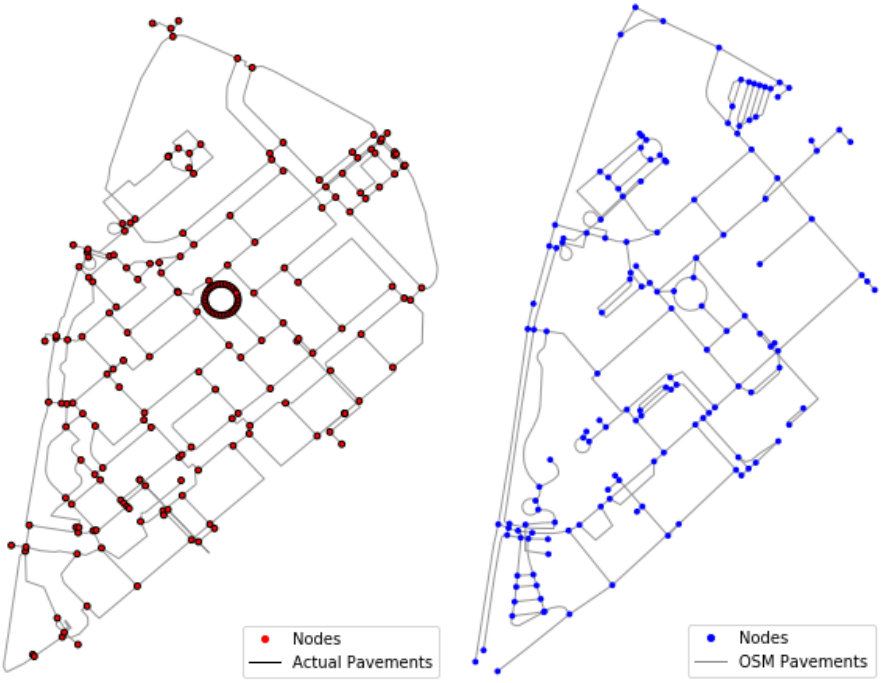


*Figure 3.10(a) (Left) – The reference network (the block **reference data** in Figure 3.7)*
*Figure 3.10(b) (Right) – The OSM (VGI) network (the block **VGI data** in Figure 3.7)*

3.5.3. Algorithm development and system development

After exporting the shapefiles, all the data from the reference network are ready. A principle algorithm has been created and has been implemented as the analysis script. The analysis script has been developed for obtaining the VGI data and the analysis result. The user shall configure their parameters (e.g. name of files) at the beginning. Figure 3.11 illustrates the developed algorithm.

3.5.3.1. Data loading for OSM

From the proposed solution, the analysis scripts will download the VGI data using the OSMnx and the given **theSite** polygon shapefile.

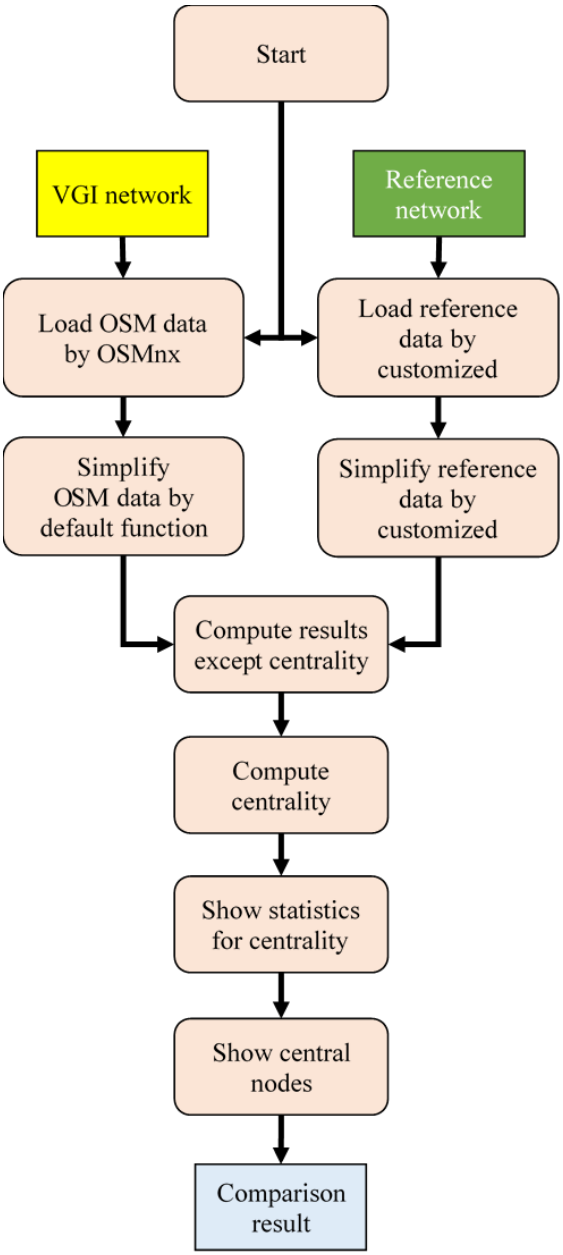


*Figure 3.11 – The algorithm of the analysis script (the **Compare** block in Figure 3.7)*

Since the topology is complex for original OSM data, OSMnx has provided a function to simplify the topology of the OSM data. The simplified topology would retain the geometry but eliminate redundant nodes, such as pseudo-nodes. This dissertation used the original function to simplify the OSM topology. Figure 3.10(b) illustrated the OSM pedestrian data downloaded and simplified by OSMnx.

3.5.3.2. System customisation for reference network

The reference network shall be simplified too in order to generate comparable results. However, the OSMnx did not intend to simplify non-OSM source data. Hence, this dissertation have overloaded the critical function in loading function (***read_shp*** ➔ ***read_shp_with_nodes***), the class for holding the network (***DiGraph*** ➔ ***MyDiGraph***), and the simplify functions (***simplify_graph*** ➔ ***simplify_simple_graph***; ***plot_graph*** ➔ ***plot_graph_simple_graph***). Appendix A has highlighted the changes in the source code. To sum up the changes, ***read_shp_with_nodes*** and ***MyDiGraph*** will only accept nodes that exist in a separate shapefile; while the ***MyDiGraph*** will also add attributes for the simplify functions. The ***simplify_simple_graph*** and ***plot_graph_simple_graph*** have modified the OSMnx the assumption of multi-graph to simple graph.

3.5.3.3. Result calculation and visualisation

After both networks have been simplified, the analysis script will compute the metric in Table 3.4 with accordance to the algorithm Figure 3.11.

First, the code segment from Appendix A will compute the non-centralities metrics:

```
# Now calculate selected metrics other than centrality

graph_osm_dir = graph_osm_copy.to_directed()
graph_ref_dir = graph_ref_copy.to_directed()

graph_osm_analysis = graph_osm_copy.to_undirected()
graph_ref_analysis = graph_ref_copy.to_undirected()

graph_osm_simple = nx.Graph(graph_osm_copy)
graph_ref_simple = nx.Graph(graph_ref_copy)

# Average Degree Connectivity
print("Average Degree Connectivity for OSM:
"+str(nx.average_degree_connectivity(graph_osm_sim-
ple)))
print("Average Degree Connectivity for REF:
"+str(nx.average_degree_connectivity(graph_ref_sim-
ple)))

# Average Path Length
print("Average Path Length for OSM: "+str(max([nx.av-
erage_shortest_path_length(C) for C in nx.con-
nected_component_subgraphs(graph_osm_simple)])))
print("Average Path Length for REF: "+str(max([nx.av-
erage_shortest_path_length(C) for C in nx.con-
nected_component_subgraphs(graph_ref_simple)])))

# Clustering Coefficient
print("Clustering Coefficient for OSM: " +
str(nx.transitivity(graph_osm_simple)))
print("Clustering Coefficient for REF: " +
str(nx.transitivity(graph_ref_simple)))

# Efficiency
print("Efficiency for OSM: " + str(nx.global_effi-
ciency(graph_osm_analysis)))
print("Efficiency for REF: " + str(nx.global_effi-
ciency(graph_ref_analysis)))

# Hierarchy
print("Flow hierarchy for OSM: " + str(nx.flow_hierar-
chy(graph_osm_dir)))
print("Flow hierarchy for REF: " + str(nx.flow_hierar-
chy(graph_ref_dir)))
```

Then, the script will compute the centrality metrics. The three centralities would be summed up as a final centrality value. The rationale of summing up all of them is to consider all the centralities; the script will compute the statistics for the final value in the following extract from Appendix A:

```
# Now calculate centrality

# Degree Centrality
deg_centrality_osm = nx.degree_central-
ity(graph_osm_analysis)
deg_centrality_ref = nx.degree_central-
ity(graph_ref_analysis)

# Betweenness Centrality
btwn_centrality_osm = nx.betweenness_central-
ity(graph_osm_analysis)
btwn_centrality_ref = nx.betweenness_central-
ity(graph_ref_analysis)

# Closeness Centrality
close_centrality_osm = nx.closeness_central-
ity(graph_osm_analysis)
close_centrality_ref = nx.closeness_central-
ity(graph_ref_analysis)

# ECDF for centrality. Use for comparison
deg_centrality_osm_ecdf = ECDF([v for k, v in deg_cen-
trality_osm.items()])
btwn_centrality_osm_ecdf = ECDF([v for k, v in
btwn_centrality_osm.items()])
close_centrality_osm_ecdf = ECDF([v for k, v in
close_centrality_osm.items()])

deg_centrality_ref_ecdf = ECDF([v for k, v in deg_cen-
trality_ref.items()])
btwn_centrality_ref_ecdf = ECDF([v for k, v in
btwn_centrality_ref.items()])
close_centrality_ref_ecdf = ECDF([v for k, v in
close_centrality_ref.items()])

# Summarize statistics
centrality_osm = dict()
centrality_percentile_osm = dict()
for k, v in deg_centrality_osm.items():
    centrality_osm[k] = v
    centrality_percentile_osm[k] = deg_central-
ity_osm_ecdf(v)
for k, v in btwn_centrality_osm.items():
    if (centrality_osm[k] is not None):
        centrality_osm[k] = centrality_osm.get(k) + v
```

```
        centrality_percentile_osm[k] = centrality_per-
centile_osm.get(k) + btwn_centrality_osm_ecdf(v)
    else:
        centrality_osm[k] = v
        centrality_percentile_osm[k] = btwn_central-
ity_osm_ecdf(k)
for k, v in close_centrality_osm.items():
    if (centrality_osm[k] is not None):
        centrality_osm[k] = centrality_osm.get(k) + v
        centrality_percentile_osm[k] = centrality_per-
centile_osm.get(k) + close_centrality_osm_ecdf(v)
    else:
        centrality_osm[k] = v
        centrality_percentile_osm[k] = close_central-
ity_osm_ecdf(v)

centrality_ref = dict()
centrality_percentile_ref = dict()
for k, v in deg_centrality_ref.items():
    centrality_ref[k] = v
    centrality_percentile_ref[k] = deg_central-
ity_ref_ecdf(v)
for k, v in btwn_centrality_ref.items():
    if (centrality_ref[k] is not None):
        centrality_ref[k] = centrality_ref.get(k) + v
        centrality_percentile_ref[k] = centrality_per-
centile_ref.get(k) + btwn_centrality_ref_ecdf(v)
    else:
        centrality_ref[k] = v
        centrality_percentile_ref[k] = btwn_central-
ity_osm_ecdf(v)
for k, v in close_centrality_ref.items():
    if (centrality_ref[k] is not None):
        centrality_ref[k] = centrality_ref.get(k) + v
        centrality_percentile_ref[k] = centrality_per-
centile_ref.get(k) + close_centrality_ref_ecdf(v)
    else:
        centrality_ref[k] = v
        centrality_percentile_ref[k] = close_central-
ity_osm_ecdf(v)

# Prepare to show the result
l_centrality_osm = [v / 3 for k, v in central-
ity_osm.items()]
l_centrality_osm_ecdf = ECDF(l_centrality_osm)
l_centrality_osm_sum_ecdf = [l_centrality_osm_ecdf(v)
for v in l_centrality_osm]
l_centrality_percentile_osm = [v / 3 for k, v in cen-
trality_percentile_osm.items()]

l_centrality_osm.sort(reverse=True)
l_centrality_osm_high = l_centrality_osm[:len(l_cen-
trality_osm) // 10]

l_centrality_percentile_osm = np.array(l_central-
ity_percentile_osm)
l_centrality_percentile_osm[::-1].sort()

l_centrality_ref = [v / 3 for k, v in central-
ity_ref.items()]
l_centrality_ref_ecdf = ECDF(l_centrality_ref)
l_centrality_percentile_ref = [v / 3 for k, v in cen-
trality_percentile_ref.items()]
```

```
l_centrality_ref.sort(reverse=True)
l_centrality_ref_high = l_centrality_ref[:len(l_cen-
trality_ref) // 10]

l_centrality_percentile_ref = np.array(l_central-
ity_percentile_ref)
l_centrality_percentile_ref[::-1].sort()
```

Since the three centralities in Table 3.4 is a node-based metric, the script will also display the distribution of two networks. The script will plot graphs on the distribution of the centralities on the following segment extracted from Appendix A:

```
# Figure: sum of three centralities
plt.plot(l_centrality_osm, label='OSM')
plt.plot(l_centrality_ref, label='Actual')

plt.legend(loc='upper right')
plt.grid(True, color=(0.85,0.85,0.85))
plt.title("Sum of the Three Centralities Measures")
plt.xlabel("n-th most central nodes")
plt.ylabel("Sum of the centralities")

#CDF of summing three centralities
plt.hist(l_centrality_osm, normed=True, cumula-
tive=True, label='OSM',
         histtype='stepfilled', color=(0,0,1,0.75))
plt.hist(l_centrality_ref, normed=True, cumula-
tive=True, label='Actual',
         histtype='stepfilled', color=(1,0,0,0.4))
#legends = [mpl.lines.Line2D([],[], color='b' la-
bel='OSM'), mpl.lines.Line2D([],[], color='r' la-
bel='Actual')]
plt.legend(loc='upper left')
plt.title("Cumulative Distribution of Centralities
Measures")
plt.xlabel("Sum of the centralities")
plt.ylabel("Probability")
```

Finally, the central nodes will be visualised on the graph. The operator could analysis the execution results generated from the code segment above.

The system has been customised and implemented. In this dissertation, a trial execution has been accomplished using a manually-digitised network conforming with Section 3.5.1, and the results would be discussed in the next section.

4. RESULTS AND ANALYSIS

This dissertation has described how to execute the procedures to evaluate the difference between the two networks. In this section, the case study results of the implementation will be introduced. There are three major categories of the results in this case study. The implication of those results will be discussed and analysed.

Eight metrics is being selected in the study of topological accuracy. Table 3.4 listed the eight evaluation metrics. The centralities analysis are individual node-based assessment, which is different from other summarisation based characters. Hence, in the following subsections, we would list three elemental compositions of our results and analysis based on the results.

4.1. Results analysis for summary-based metrics

There are five characters of a graph belongs to "summary-based metrics". They are "average degree connectivity", "ASPL", "clustering coefficient (transitivity)", "efficiency" and "flow hierarchy". Table 4.1 summarised the numerical results of these five evaluations.

Metric	VGI	Reference
Average degree connectivity	1: 2.76 2: 2.47 3: 2.91 4: 3.05	1: 2.4 2: 2.14 3: 2.72 4. 2.61
ASPL	8.561	9.5
Transitivity	0.0331	0.00865
Efficiency	0.159	0.0792
Flow hierarchy	0*	0.738*

Table 4.1 – Summarisation-based metrics result

The first metric is the average degree connectivity. It shows the average degree of the nearest neighbour. For example, VGI's 1:2.76 means that the average 2.76 degrees are discovered from the neighbour of all nodes having degree 1. From the table, we could observe that VGI's average degree connectivity for degree 1 to 2 is dropped from 2.76 to 2.47. Then the value for degree 2 to 4is increased from 2.47 to 3.05. For reference network, although it decreases and increases zig-zag from 2.4 to 2.61; however, all degree 1 to 4 having a lower average nearest neighbour degree. This result shows that the reference pavements are less connected than the OSM pavements.

The second metric is ASPL, which demonstrates the average shortest path length to reach the farthest elements. A higher value indicates the farther connection or less connected graph than the lower value. The ASPL of OSM network is 8.561, which is lower than the reference network's 9.5. It also demonstrates that OSM is more connected than the reference network.

The third metric is transitivity, which is a probability that two random nodes are connected. Higher probability means a well-connected graph (small-world phenomena). The result from OSM is 3.31%, while the results of the reference are 0.865%. This result indicates reference networks having less probability that two random nodes are connected; hence a less connected graph.

The fourth metric is efficiency. Higher efficiency means easier to reach other parts of the graph. The result of OSM is 0.159, which is larger than the reference's 0.0792. Hence, it demonstrates that the OSM network is

faster to reach the other parts of the network, hence a better-connected network topology.

The first four metrics have all indicated that the OSM are better-connected than the reference network. The better connection may be explained by the result of metric 5 – "flow hierarchy". Flow hierarchy measures the percentage of the edges not in cycles in a directed graph. OSM shows 0 per cent, which is significantly lower than the reference's 73.8 per cent. The result indicates that the OSM network is fully-connected with each other. The very high flow hierarchy value of the reference graph also indicates that there are very few cycles in the reference network.

To sum up the observations, OSM pavement networks shows a better-connected graph than the reference pavements networks. The metric of "flow hierarchy" indicates that there are not many cycles in the reference network. After investigating the issues, this dissertation discovered that many connections to the boundary of the site in the reference network, are footbridges and road crossing. In contrast, the OSM has very few of these only. Also, the reference is expected to have disconnected subgraph. Both contribute to the discrepancies between OSM and reference network. Those difference are moderately critical to some network analysis function, such as routing, could be wrong if the network is connected unexpectedly. Hence these discrepancies on OSM pavements introduces inaccuracies on topological quality and external logical inconsistency. The current accuracy of OSM data might be insufficient to support task needed high topological accuracy in the site.

In order to ease the problems of current VGI data, the development team and community could contribute to the VGI platform, in private use or open to the public, in order to maintain better topological accuracy in VGI data. The popularity of VGI platform is due to it is open for the public's contributions. Such problems could be eased quickly with public engagement.

Otherwise, the developers shall consider alternative solutions available. In a worse scenario, the development team may require manually create a comparable network if all other sources are not available. However, the cost could be huge if a sizable network is being created.

A specific problem could be discovered in our case study. From the introduction, one could understand that one of the underlying problems of the case study is "no other data open" but not "no other data available". For the case study, the local government has promised a "walk smart" initiative. The authority shall consider opening the data they held. It is absurd if an authority wants to co-create a smart community but critical data infrastructures for the public, such as the pedestrian network format, are missing.

4.2. Results analysis for centrality measures

In contrast with numerical values, the centrality assessment would assess and assign a value for each node. Hence, it provides more information than numerical values. The centrality measures in this dissertation refer to summing up the three centralities used in Table 3.4. The methodology described the calculation.

First, this dissertation has plotted two graphs in Figure 4.1. Figure 4.1(a) delineated the graph of sorting the centrality value in descending order, then plot the graph by the value against the order of the value. While Figure 4.1(b) plotted the cumulative distribution of the centrality measure defined in this dissertation.

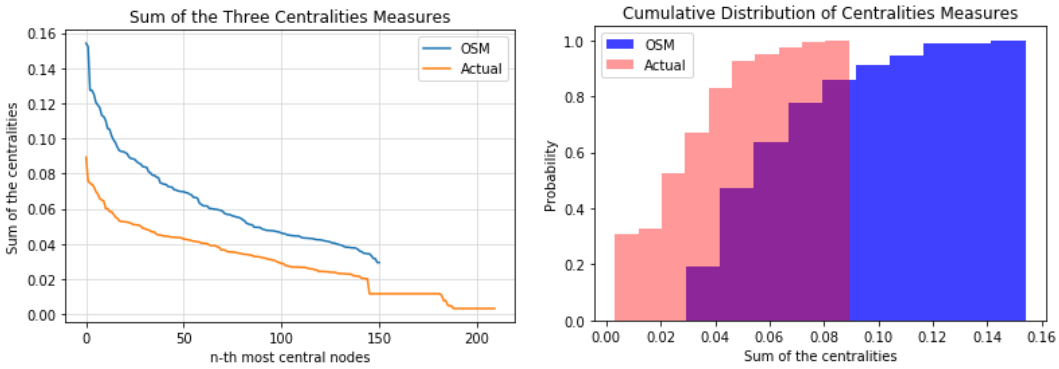


Figure 4.1(a) (Left) – Sum of the centralities versus the n-th most central nodes.
Figure 4.1(b) (Right) – Cumulative distribution chart for the centrality measure.

In Figure 4.1(a), the blue line refers to the OSM data while the orange line refers to the reference graph. From the graph, the OSM's centrality measure reaches its highest at about 0.156, then dropped to 0.03 gradually. In contrast, the highest value of the actual network is about 0.09 only. The value drops to 0.022, but flatter than the OSM value. Afterwards, there are two flat low-lands around the 145th most central nodes to the 175th most central node and about 185th most central node to the least central nodes. Figure 4.1(a) demonstrates three critical issues. First, the distribution of the centrality value is somewhat different between the OSM pavement and the reference network. Second, the OSM's centrality measure is always higher than the reference's one. This observation could be explained by OSM network are

more connected than the reference network from a previous analysis in Section 4.1. Finally, the OSM network is not complete since it has fewer nodes than the reference pavement.

The cumulative distribution graph in Figure 4.1(b) is intended to show the slope of the cumulative measures. From the graph, the increasing speed of the “blue” OSM centrality measure is decreasing consistently. This decrement leads to a curve similar to a circular arc on the top of the bar chart. However, the “red” reference network shows an inconsistent change. The graph demonstrated that the rate of change in the two centrality measures is different.

Despite the differences in distribution, the central nodes also have different, as demonstrated in Figure 4.2. Figure 4.2 highlighted the top 10% most central nodes in the OSM network, as well as the reference network. In Figure 4.2(a), there are three significant circled areas (orange circle) for the central nodes in OSM pavement. Although in figure 4.2(b) demonstrated five smaller areas (blue circle) for reference network. There are three significant observations in these two figures.

First, although two figures look not quite similar by cycled area, indeed the most central nodes on both figures show the most central node are located around The Urban Council Centenary Garden (red circle in Figure 4.2). The finding reveals that the OSM network can approximate the central nodes, despite the differences exists.

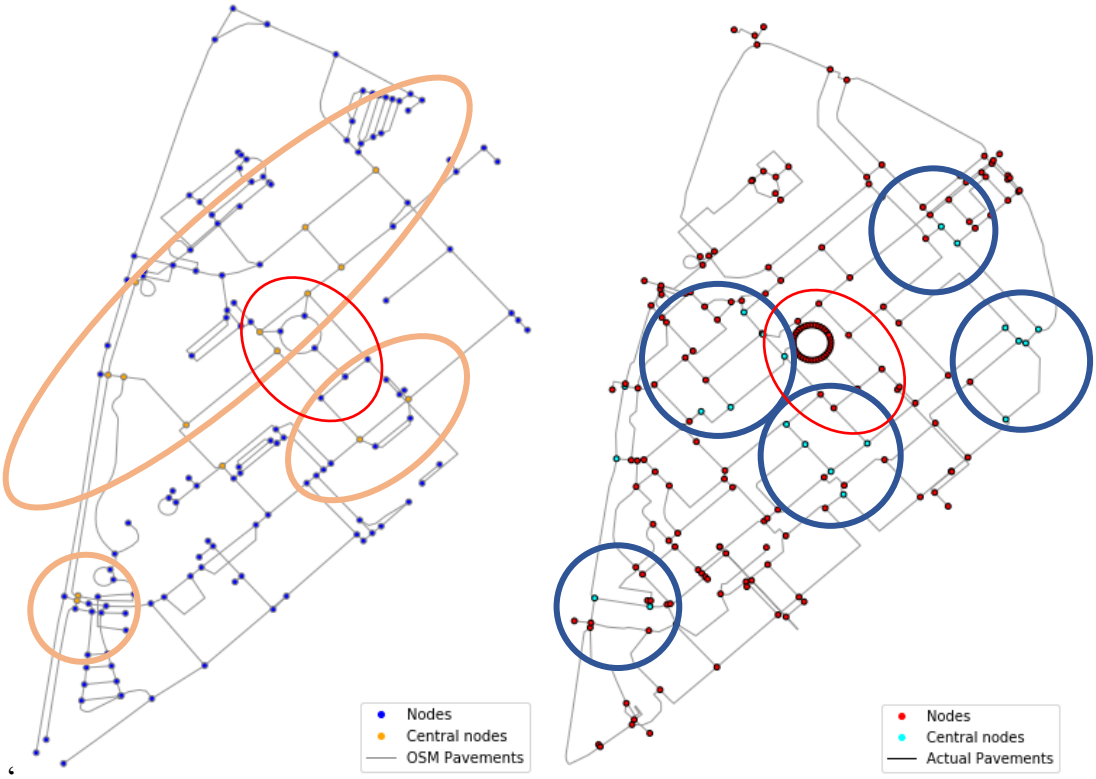


Figure 4.2(a) (Left) – The top 10% most central nodes in the OSM data.
Figure 4.2(b) (Right) – The top 10% most central nodes in the reference data.

Second, according to field observations, OSM can demonstrate the paths with the highest pedestrian flow by central nodes only. The paths around The Urban Council Centenary Garden (red circle in Figure 4.2) are the most visited region in field observations. The reference network's topology shows a less-visited central path than the OSM's central paths. However, the flow value is also critical when calculating the central nodes by weighting each node and edges. Hence, the central nodes accuracy deserves more research with the account of the flow model.

Third, missing nodes in OSM are influencing the computation of central nodes. In Figure 4.2(b), the left bottom part (south-east) is accounted as central nodes, while they are not central nodes in Figure 4.2(a). Coincidentally or not, there is a small missing area from the OSM data in Figure 4.2(a). Hence, it may show that the missing data of OSM might affect the topological accuracy and hence, external logical consistency. However, as our problem definition stated that, completeness is a critical measure in overall accuracy. The general procedures shall already assess the completeness; therefore, the completeness metric shall already account the inaccuracies.

To sum up, those observations, although OSM data are incomplete for a small area and influencing the results, the OSM data in this study demonstrates a close approximation to the actual centrality assessment. The centrality measures used in the study are node-based topological structures assessment. The most central (and hence significant) nodes are more critical than the other nodes. The result demonstrates that the centrality assessment pinpoints the area surrounding the same area. The results may implicate that the OSM data has a capable topology for the significant nodes, such as centrality assessment.

However, the missing data and shifted position also indicated that using the OSM data requires extreme care. Some less-visited region may have significant missing and inaccuracies on topology structures. Hence, it is preferred to have sampled on OSM data and verify the OSM data with a reference network. Also, external logical consistency relies on the completeness of the OSM data; hence, a higher weight in assessing usability are welcomed.

5. CONCLUSION

This section is going to summarise the project described in this dissertation. The key findings will be highlighted once again. Finally, this dissertation will review the limitation and recommendation for future studies.

5.1. Project summarisations

A comprehensive literature review has discovered that past studies have omitted a crucial part of logical consistency – the external logical consistencies. Any logical consistency that is compared with an external source would be external logical consistency. It is also discovered that VGI is a common data source for geographic information like network data format. However, how the network produced by OSM portrays the ground truth is unclear. Indeed, those accuracy evaluation requires a reference network; hence, it is an external logical consistency. This dissertation has the aim to develop procedures to evaluate the external logical consistency and tried to validate the approach by a case study on the site at East Tsim Sha Tsui, Hong Kong.

The methodology explained the requirements of the project and described how to fulfil the requirement by a proof-of-concept implementation. The requirement on the hardware, software and data are listed and modelled explicitly. The hardware and software usage is designed with simplicity, system interoperability, data integration and platform-independent such that board and inexperienced user of VGI could also replicate the result. The algorithm selected is based on practicability and effectiveness. Afterwards,

a case study, that is, the implementation, are described in details for the analyst's decision to adopt or modify. Finally, the results of the case study have been documented and analysed.

5.2. Key findings

There are a few significant key findings in the case study.

First, the OSM network in the case study is more connected than expected by metrics. The primary reasons would be an unexpected cycle around the boundary of the site, as well as no expected disconnected subgraphs. Some part that is not intentionally connected will lead to inaccuracies in the application requires an entirely correct database, like routing and navigation. In order to resolve the problem on use cases of navigation, the user of the VGI needed to commit changes into the VGI data. An expensive but more accurate way is to maintain its network model. However, as a city promote "walk smartly", it is unacceptable to publish no open network model even the authorities have one.

Second, OSM network generally portrays similar central nodes around the central area despite the differences in their distributions. The value of centrality and their distribution are different. However, the centrality analysis shows that the OSM can portray the central nodes around the centre of the pavement study. The result reveals that OSM could be an acceptable source for analysis of the major skeleton of the city. However, the incompleteness of the OSM requires extreme care. This dissertation recommends one could perform sampling and create network model in samples for obtaining better confidences on the analysis result.

5.3. Future improvement and recommendation

There are a few assumptions in this dissertation. First, in the algorithm development stage, we assumed that the reference network is the unquestionable ground truth. However, as the literature review states that, the network models depended on the geometry of nodes and edges. Hence, the result could be further improved by introducing uncertainties in the reference network.

Second, in the result analysis stage, we assumed the completeness would be corrected in another part of the research. It is a true statement if the research aims to evaluate the ISO 19157 usability of VGI data. However, in performing other types of research other than quality validation, the completeness may be overlooked. Hence, in future studies, an external logical consistency indicator fulfilling the ISO 19157 standards might be preferred for "take and go" measurements.

Furthermore, our network models have not considered the flow-entity model in conventional network analysis. Flow value or impedance could be implemented as a weight of the nodes or edges. Those weighting could result in different analysis outcome. Hence, a future study could perform a survey or a census in order to embed the flow entity model in the analysis network structure for GIS.

Finally, the study could be further refined and validated by performing a multi-location study. The current study used a manually created network as a reference network. Performing studies on this procedure on more site and more data could further validate and verify the results of this dissertation.

6. REFERENCES

Antoniou, V., Morley, J., & Hakley, M. (2010). Web 2.0 geotagged photos: Assessing the spatial dimension of the phenomenon. *Geomatica*, *64*(1), 99-110.

Antoniou, V., Skipeliti, A. (2015). Measures and indication of VGI quality: an overview. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science*, II(3/W5), 345-351.

Bavelas, A. (1948). A mathematical model for group structures. *Human Organization*, 7, 16-30.

Bell, M. G., & Iida, Y. (1997). *Transportation network analysis*. New York: J. Wiley.

Boeing, G. (2017). OSMnx: New methods for acquiring, constructing, analysing, and visualizing complex street networks. Computers, Environment and Urban Systems, 65, 126-139.

Boeing, G. (2018). A multi-scale analysis of 27,000 urban street networks: every US city, town, urbanized area, and Zillow neighborhood. *Environment and Planning B: Urban Analytics and City Science*. doi:10.1177/2399808318784595

Brovelli, M. A., Minghini, M., Molinari, M. E. (2016). An automated GRASS-based procedure to assess the geometrical accuracy of the OpenStreetMap Paris road network. *XXIII ISPRS Congress*, XLI(B7), 919-925.

Carter, M. W., & Price, C. C. (2001). *Operations research: a practical introduction*. Boca Raton: CRC Press.

Chartrand, G. (1985). *Introductory graph theory*. New York: Dover.

Chrisman, N. R. (1991). The error component in spatial data. In D. J. Maguire, M. F. Goodchild, & D. W. Rhind (Ed.), Geographic Information Systems (pp. 165-174). Harlow: Longman Scientific and Technical.

Cipeluch, B., Jacob, R., Winstanley, A., & Mooney, P. (2010). Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. *In Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Science*. Retrieved from http://ww.cs.nuim.ie/

Cohen, B. (2014). *The smart cities in the World: methodology*. Retrieved from https://www.fastcompany.com/3038818/the-smartest-cities-in-the-world-2015-methodology

Cohen, B. (2015). *The 3 generations of smart cities*. Retrieved from https://www.fastcompany.com/3047795/the-3-generations-of-smart-cities

Corcoran, P., Mooney, P., & Winstanley, A. (2010, April). Topological consistent generalization of OpenStreetMap. Paper presented at GISRUK 2010: GIS Research UK 18th Annual Conference, London. Retrieved from https://pdfs.semanticscholar.org/a91f/c9a43981280877d2f5802dc2057e716433eb.pdf?_ga=2.40151777.2126566915.1563876614-986764308.1563876614

Curtin, K. M. (2007). Network analysis in Geographic Information Science: Review, assessment and projections. *Cartography and Geographic Information Science*¸ *34*(7), 103-111.

Fischer, M. M. (2006). GIS and network analysis. *Spatial Analysis and GeoComputation: Selected Essays* (pp. 43-60). Heidelberg: Springer Berlin.

Fonte, C.C., Antoniou, V., Bastin, L., Estima, J., Arsanjani, J. J., Bayas, J-C. L., See, L., & Vatseva, R. (2017). Assessing VGI data quality. In G. Foody, L. See, S. Fritz, P. Mooney, A-M. Olteanu-Raimond, C. C. Fonte, & V. Antoniou (Ed.), Mapping and the Citizen Sensor (pp. 137-163). London: Ubiquity Press. doi:10.5334/bbf.g

Ford, Jr., L. R., & Fulkerson, D. R. (1962). *Flows in networks*. Retrieved from https://www.rand.org

Forghani, M., Delavar, M. R. (2014). A Quality Study of the OpenStreetMap Dataset for Tehran. *ISPRS International Journal of Geo-Information*, 3(2), 750-763.

Gera, R., Alonso, L., Crawford, B., House, J., Mendez-Bermudez, J. A., Knuth, T. (2018). Identifying network structure similarity using spectral graph theory. *Applied Network Science*, doi:10.1007/s41109-017-0042-3

Girres, J. F., & Touya, G. (2010). Quality assessment of the French OpenStreetMap dataset. *Transactions in GIS*, 14(4), 435-459.

Girres, J-F, & Touya, G. (2010). Quality assessment of the French OpenStreetMap dataset. Transactions in GIS, 14(4). 435-459.

Goodchild, M. (2007). *Citizens as censors: the World of volunteered geography* [PDF file]. Retrieved from www.ncgia.ucsb.edu/projects/vgi/docs/position/Goodchild_VGI2007.pdf

Graser, A., Straub, M., & Dragaschnig, M. (2015). Is OSM good enough for vehicle routing? A study comparing street networks in Vienna. In G. Gartner, & H. Huang (Ed.), *Progress in Location-Based Services 2014* (pp. 3-17). Switzerland: Springer

Haklay, M. (2010). How good is volunteered geographic information? A comparative study of OpenStreetMap and Ordnance Survey datasets. *Environmental and Planning B: Urban Analytics and City Science*, 37. 682-703.

Hakley, M. (2010). How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets. Environment and Planning B: Urban Analytics and City Science, 37(4), 682-703.

Harary, F. (1994). *Graph theory*. HA: Addison Wesley.

Helbich, M., Amelunxen, C., Neis, P., & Zipf, E. (2012). Comparative spatial analysis of positional accuracy of OpenStreetMap and proprietary geodata. In T. Jekel, A. Car, J. Strobl, & G. Griesebner (Ed.)., *Proceedings of the Geoinformatics Forum Salzburg*. Retrieved from: https://www.geog.uni-heidelberg.de/

Higgins, C. D. (2019). A 4D spatio-temporal approach to modelling land value uplift from rapid transit in high density and topographically-rich cities. Landscape and Urban Planning, doi:10.1016/j.landurbplan.2018.12.011

Higgins, C. D. (2019). A 4D spatio-temporal approach to modelling land value uplift from rapid transit in high density and topographically-rich cities. *Landscape and Urban Planning*, 185(May 2019), 68-82.

Innovation and Technology Bureau. (2017). Smart city blueprint for Hong Kong [PDF file]. Retrieved from https ://www.smartcity.gov.hk/doc/HongKongSmartCityBlueprint(EN).pdf

Innovation and Technology Bureau. (2017). *Smart city blueprint for Hong Kong* [PDF file]. Retrieved from https://www.smartcity.gov.hk/doc /HongKongSmartCityBlueprint(EN).pdf

Jiang, B. (2007). A topological pattern of urban street networks: universality and peculiarity. *Physics A*, 384(2007), 647-655.

Jiang, B., & Claramunt, C. (2004). Topological analysis of urban street networks. *Environment and Planning B: Planning and Design*. doi:10.1068/b306.

Jokar Arsanjani, J., & Bakillah, M. (2015). Understanding the potential relationship between the socio-economic variables and contributions to OpenStreetMap. *International Journal of Digital Earth*, *8*(11), 861-876.

Kounadi, O. (2009). *Assessing the quality of OpenStreetMap data* (Master's thesis, University College of London). Retrieved from https://www.researchgate.net /publication/264553789_Assessing_the_Quality_of_OpenStreetMap_Data

Longley, P. A., Goodchild, M. F., Maguire, D. J., & Rhind, D. W. (2005). *Geographic information systems and science*. Chichester: John Wiley & Sons.

Luo, J., & Magee, C. L. (2011). Detecting evolving patterns of self-organizing networks by flow hierarchy measurement. *Complexity, 16*(6), 53-61.

Lupien, A. E., Moreland, W. H., & Dangermod, J. (1987). Network analysis in Geographic Information Systems. Photogrammetric Engineering and Remote Sensing, 53(10), 1417-1421.

Maguire, D. J. (1991). An overview and definition of GIS. In D. J. Maguire, M. F. Goodchild, & D. W. Rhind (Ed.), *Geographic Information Systems* (pp. 9-20). Harlow: Longman Scientific and Technical.

Maguire, D. J., Goodchild, M. F., & Rhind, D. W. (1991). Geographic Information Systems. Harlow: Longman Scientific and Technical.

Mapbox. (n.d.) *The OpenStreetMap data model*. Retrieved from https://labs.mapbox.com/mapping/osm-data-model

Millar, H. J., & Shaw, S. -L. (2001). Geographic information systems for transportation. New York: Oxford University Press.

Monteiro, E. S., Fonte, C. C., de Lima, J. L. (2018). Improving the positional accuracy of drainage networks extracted from global digital elevation models using OpenStreetMap data. *Journal of Hydrology and Hydromechanics*, 66(3), 285-294.

Office of the Chief Executive, The Government of the Hong Kong Special Administrative Region. (2016). *The 2016 policy address* [PDF file]. Retrieved from

https://www.policyaddress.gov.hk/2016/eng/pdf/PA2016.pdf

Olbricht, R. M. (2015). Data retrieval for small spatial regions in OpenStreetMap. In J. Jokar Arsanjani, A. Zipf, P. Mooney, & M. Helbich (Ed.), *OpenStreetMap in GIScience* (pp. 101-122). Switzerland: Springer.

Opensaw, S. (1991). Developing appropriate spatial analysis methods for GIS. In D. J. Maguire, M. F. Goodchild, & D. W. Rhind (Ed.), *Geographic Information Systems* (pp. 9-20). Harlow: Longman Scientific and Technical.

Ordnance Survey. (n.d.). *Crowd sourcing*. Retrieved from https://www.ordnancesurvey.co.uk/education-research/research/crowd-sourcing.html

Porta, S., Crucitti, P., & Latora, V. (2006). The network analysis of urban streets: a primal approach. *Environment and Planning B: Planning and Design*. doi:10.1068/b32045

Pourabdollah, A., Morley, J., Feldman, S., & Jackson, M. (2013). Towards an authoritative OpenStreetMap: conflating OSM and OS OpenData National Maps' road network. *ISPRS International Journal of Geo-Information*, 2013(2), 704-728.

Rodrigue, J. P. (2017). *The geography of transport system* (4th ed.). New York: Routledge. Retrieved from https://transportgeography.org/

Shaw, S.-L. (2010). Geographic information systems for transportation: from a static past to a dynamic future. *Annals of GIS*, *16*(3), 129-140.

Tang, Y. F. (2005). Algorithmic Development of an optimal path computation model based on topographic map features (Master's thesis, The Hong Kong Polytechnic University). Retrieved from http://ira.lib.polyu.edu.hk/bitstream/10397/3589/2/b20592929_ir.pdf

The Government of the Hong Kong Special Administrative Region. (2018). *GovHK: HKeMobility*. Retrieved from https://www.gov.hk/en/residents/transport/publictransport/hketransport.htm

U.S. Department of Transport. (2009). *Geographic information systems application for bicycle and pedestrian decision-making: Peer exchange summary report* [HTML file]. Retrieved from https://www.gis.fhwa.dot.gov/documents/GIS_BikePed_Peer_rpt.htm

Upchurch, C. (2004). Using GIS to generate mutually exclusive service areas linking travel on and off a network. *Journal of Transport Geography*, *12*(1), 23-33.

USGS. (n.d.). *Volunteered geographic information (VGI)*. Retrieved from https://www.usgs.gov/core-science-systems/ngp/cegis/vgi

Wilson, R. J. (1996). *Introduction to graph theory* (4th ed.). Harlow: Addison Wesley Longman.

Zhou, P., Huang, W., & Jang, J. (2014). Validation analysis of OpenStreetMap data in some areas of China. *ISPRS Technical Commission IV Symposium*, XL-4(May), 383-391.

APPENDIX A – SOURCE CODE OF THE ANALYSIS SCRIPT

```
import os

# Define variables for the execution

if not 'wd' in globals():
    wd = os.getcwd()

site_area = os.path.join(wd, "OSM", "osmnx", "site_area_4326.shp")
node_ref = os.path.join(wd, "iB1000", "4326", "Ped_Junc_4326.shp")
edge_ref = os.path.join(wd, "iB1000", "4326", "Ped_Line_4326.shp")
ref_proj = 'EPSG:4326'

# Try to import related package
# If uninstalled, then try to install it
try:
    import sys
    import pip
    import time
    import math

    import fiona
    import geopandas as gpd
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import networkx as nx
    import numpy as np
    from osgeo import ogr
    import osmnx as ox
    import scipy as sc
    import shapely
    from shapely.geometry import LineString
    from shapely.geometry import Point
    from shapely.geometry import Polygon
    from statsmodels.distributions.empirical_distribution import ECDF
    from itertools import chain
    from collections import Counter
except ImportError:
    print("Required package not found.")


# Read data using OSMNX
c = fiona.open(site_area)
pol = c.next()
geom = shapely.geometry.shape(pol['geometry'])
graph_osmnx = ox.graph_from_polygon(geom, network_type='walk')
```

```
# Now extend the NetworkX class and NetworkX function.
# NetworkX is licensed through BSD 3-Clause (Revised) License.
# For license information, please refer to the License Information in the LICENSE_NetworkX.md
in License folder.
class MyDiGraph(nx.DiGraph):
    pass

    def add_edge_without_add_node(self, u_for_edge, v_for_edge, key=None, **attr):
        """Add an edge between u and v.
        The nodes u and v will be automatically added if they are
        not already in the graph.
        Edge attributes can be specified with keywords or by directly
        accessing the edge's attribute dictionary. See examples below.
        Parameters
        ----------
        u, v : nodes
            Nodes can be, for example, strings or numbers.
            Nodes must be hashable (and not None) Python objects.
        attr : keyword arguments, optional
            Edge data (or labels or objects) can be assigned using
            keyword arguments.
        See Also
        --------
        add_edges_from : add a collection of edges
        Notes
        -----
        Adding an edge that already exists updates the edge data.
        Many NetworkX algorithms designed for weighted graphs use
        an edge attribute (by default `weight`) to hold a numerical value.
        Examples
        --------
        The following all add the edge e=(1, 2) to graph G:
        >>> G = nx.Graph()   # or DiGraph, MultiGraph, MultiDiGraph, etc
        >>> e = (1, 2)
        >>> G.add_edge(1, 2)           # explicit two-node form
        >>> G.add_edge(*e)             # single edge as tuple of two nodes
        >>> G.add_edges_from( [(1, 2)] ) # add edges from iterable container
        Associate data to edges using keywords:
        >>> G.add_edge(1, 2, weight=3)
        >>> G.add_edge(1, 3, weight=7, capacity=15, length=342.7)
        For non-string attribute keys, use subscript notation.
        >>> G.add_edge(1, 2)
        >>> G[1][2].update({0: 5})
        >>> G.edges[1, 2].update({0: 5})
        """
        u, v = u_for_edge, v_for_edge
        # add nodes
        # Check if u may exist
        u_exists = False
        v_exists = False
        if u not in self._succ:
            self._succ[u] = self.adjlist_inner_dict_factory()
            self._pred[u] = self.adjlist_inner_dict_factory()
            ### WARNING: EDITED
            attribute = dict()
            attribute['x'] = u[0]
            attribute['y'] = u[1]
            self._node[u] = self.node_attr_dict_factory(**attribute)
            ### EDITED REGION END
        if v not in self._succ:
            self._succ[v] = self.adjlist_inner_dict_factory()
```

```
            self._pred[v] = self.adjlist_inner_dict_factory()
            ### WRANING: EDITED
            attribute = dict()
            attribute['x'] = v[0]
            attribute['y'] = v[1]
            self._node[v] = self.node_attr_dict_factory(**attribute)
            ### EDITED REGION END
        # add the edge
        datadict = self._adj[u].get(v, self.edge_attr_dict_factory())
        datadict.update(attr)
        self._succ[u][v] = datadict


def read_shp_with_node(paths, simplify=False):
    """Generates a networkx.DiGraph from shapefiles. Point geometries are
    translated into nodes, lines into edges. Coordinate tuples are used as
    keys. Attributes are preserved, line geometries are simplified into start
    and end coordinates. Accepts a single shapefile or directory of many
    shapefiles.

    "The Esri Shapefile or simply a shapefile is a popular geospatial vector
    data format for geographic information systems software [1]_."

    Parameters
    ----------
    paths : list
       a list of two shapefiles.
       paths[0] should be the node shapefile of reference network.
       paths[1] shall be the shapefile of edges.

    simplify:  bool
        If ``True``, simplify line geometries to start and end coordinates.
        If ``False``, and line feature geometry has multiple segments, the
        non-geometric attributes for that feature will be repeated for each
        edge comprising that feature.

    Returns
    -------
    G : MyDiGraph extended from DiGraph graph

    Examples
    --------
    >>> G=nx.read_shp_with_node('node.shp', 'edge.shp') # doctest: +SKIP

    References
    ----------
    .. [1] http://en.wikipedia.org/wiki/Shapefile
    """
    try:
        from osgeo import ogr
    except ImportError:
        raise ImportError("read_shp requires OGR: http://www.gdal.org/")

    if not isinstance(paths, list):
        return

    ### WARNING: EDITED
    net = MyDiGraph()
    ### Edit region end

    ### WARNING EDITED:
    for path in paths:
        ### Edited region end
```

```
        shp = ogr.Open(path)
        for lyr in shp:
            fields = [x.GetName() for x in lyr.schema]
            for f in lyr:
                flddata = [f.GetField(f.GetFieldIndex(x)) for x in fields]
                g = f.geometry()
                attributes = dict(zip(fields, flddata))
                attributes["ShpName"] = lyr.GetName()
                if g.GetGeometryType() == 1:  # point
                    point = g.GetPoint_2D(0)
                    attributes['x'] = point[0]
                    attributes['y'] = point[1]
                    net.add_node(point, **attributes)
                if g.GetGeometryType() == 2:  # linestring
                    last = g.GetPointCount() - 1
                    attributes['length'] = g.Length()
                    if simplify:
                        attributes["Wkb"] = g.ExportToWkb()
                        attributes["Wkt"] = g.ExportToWkt()
                        attributes["Json"] = g.ExportToJson()
                        net.add_edge(g.GetPoint_2D(0), g.GetPoint_2D(last), **attributes)
                    else:
                        # separate out each segment as individual edge
                        for i in range(last):
                            # Rounded to 3 decimal
                            pt1 = g.GetPoint_2D(i)
                            pt2 = g.GetPoint_2D(i + 1)
                            segment = ogr.Geometry(ogr.wkbLineString)
                            segment.AddPoint_2D(pt1[0], pt1[1])
                            segment.AddPoint_2D(pt2[0], pt2[1])
                            attributes["Wkb"] = segment.ExportToWkb()
                            attributes["Wkt"] = segment.ExportToWkt()
                            attributes["Json"] = segment.ExportToJson()

                            net.add_edge_without_add_node(pt1, pt2, **attributes)
    return net


# Now redefine the function in OSMnx for reading our dataset.
# WARNING: The source code may edited in non-specified region
#          especially for reference a module imported.

# NetworkX is licensed under MIT License.
# For license information, please refer to the License Information in the LICENSE_OSMnx.md in
License folder.
def simplify_simple_graph(G, strict=True):
    """
    Simplify a graph's topology by removing all nodes that are not intersections
    or dead-ends.
    Create an edge directly between the end points that encapsulate them,
    but retain the geometry of the original edges, saved as attribute in new
    edge.
    Parameters
    ----------
    G : networkx digraph
    strict : bool
        if False, allow nodes to be end points even if they fail all other rules
        but have edges with different OSM IDs
    Returns
    -------
    networkx multidigraph
    """
```

```
    if ox.is_simplified(G):
        raise Exception('This graph has already been simplified, cannot simplify it again.')

    ox.log('Begin topologically simplifying the graph...')
    G = G.copy()
    initial_node_count = len(list(G.nodes()))
    initial_edge_count = len(list(G.edges()))
    all_nodes_to_remove = []
    all_edges_to_add = []

    # construct a list of all the paths that need to be simplified
    paths = ox.get_paths_to_simplify(G, strict=strict)

    start_time = time.time()
    for path in paths:

        # add the interstitial edges we're removing to a list so we can retain
        # their spatial geometry
        edge_attributes = {}
        for u, v in zip(path[:-1], path[1:]):

            # there shouldn't be multiple edges between interstitial nodes
            if not G.number_of_edges(u, v) == 1:
                ox.log('Multiple edges between "{}" and "{}" found when simplifying'.for-
mat(u, v), level=lg.WARNING)
            # the only element in this list as long as above check is True
            # (MultiGraphs use keys (the 0 here), indexed with ints from 0 and
            # up)

            ### WARNING: EDITED
            edge = G.edges[u, v]
            ### Edit region ended
            for key in edge:
                if key in edge_attributes:
                    # if this key already exists in the dict, append it to the
                    # value list
                    edge_attributes[key].append(edge[key])
                else:
                    # if this key doesn't already exist, set the value to a list
                    # containing the one value
                    edge_attributes[key] = [edge[key]]

        for key in edge_attributes:
            # don't touch the length attribute, we'll sum it at the end
            if len(set(edge_attributes[key])) == 1 and not key == 'length':
                # if there's only 1 unique value in this attribute list,
                # consolidate it to the single value (the zero-th)
                edge_attributes[key] = edge_attributes[key][0]
            elif not key == 'length':
                # otherwise, if there are multiple values, keep one of each value
                edge_attributes[key] = list(set(edge_attributes[key]))

        # construct the geometry and sum the lengths of the segments
        edge_attributes['geometry'] = LineString([Point((G.nodes[node]['x'],
G.nodes[node]['y'])) for node in path])
        edge_attributes['length'] = sum(edge_attributes['length'])

        # add the nodes and edges to their lists for processing at the end
        all_nodes_to_remove.extend(path[1:-1])
        all_edges_to_add.append({'origin': path[0],
                                 'destination': path[-1],
                                 'attr_dict': edge_attributes})
```

```
    # for each edge to add in the list we assembled, create a new edge between
    # the origin and destination
    for edge in all_edges_to_add:
        G.add_edge(edge['origin'], edge['destination'], **edge['attr_dict'])

    # finally remove all the interstitial nodes between the new edges
    G.remove_nodes_from(set(all_nodes_to_remove))

    G.graph['simplified'] = True

    msg = 'Simplified graph (from {:,} to {:,} nodes and from {:,} to {:,} edges) in {:,.2f}
seconds'
    ox.log(msg.format(initial_node_count, len(list(G.nodes())), initial_edge_count,
len(list(G.edges())),
                      time.time() - start_time))
    return G


def plot_graph_simple_graph(G, bbox=None, fig_height=6, fig_width=None, margin=0.02,
                            axis_off=True, equal_aspect=False, bgcolor='w', show=True,
                            save=False, close=True, file_format='png', filename='temp',
                            dpi=300, annotate=False, node_color='#66ccff', node_size=15,
                            node_alpha=1, node_edgecolor='none', node_zorder=1,
                            edge_color='#999999', edge_linewidth=1, edge_alpha=1,
                            use_geom=True):
    """
    Plot a networkx spatial graph.
    Parameters
    ----------
    G : networkx digraph
    bbox : tuple
        bounding box as north,south,east,west - if None will calculate from
        spatial extents of data. if passing a bbox, you probably also want to
        pass margin=0 to constrain it.
    fig_height : int
        matplotlib figure height in inches
    fig_width : int
        matplotlib figure width in inches
    margin : float
        relative margin around the figure
    axis_off : bool
        if True turn off the matplotlib axis
    equal_aspect : bool
        if True set the axis aspect ratio equal
    bgcolor : string
        the background color of the figure and axis
    show : bool
        if True, show the figure
    save : bool
        if True, save the figure as an image file to disk
    close : bool
        close the figure (only if show equals False) to prevent display
    file_format : string
        the format of the file to save (e.g., 'jpg', 'png', 'svg')
    filename : string
        the name of the file if saving
    dpi : int
        the resolution of the image file if saving
    annotate : bool
        if True, annotate the nodes in the figure
    node_color : string
        the color of the nodes
    node_size : int
```

```
        the size of the nodes
    node_alpha : float
        the opacity of the nodes
    node_edgecolor : string
        the color of the node's marker's border
    node_zorder : int
        zorder to plot nodes, edges are always 2, so make node_zorder 1 to plot
        nodes beneath them or 3 to plot nodes atop them
    edge_color : string
        the color of the edges' lines
    edge_linewidth : float
        the width of the edges' lines
    edge_alpha : float
        the opacity of the edges' lines
    use_geom : bool
        if True, use the spatial geometry attribute of the edges to draw
        geographically accurate edges, rather than just lines straight from node
        to node
    Returns
    -------
    fig, ax : tuple
    """

    ox.log('Begin plotting the graph...')
    node_Xs = [float(x) for _, x in G.nodes(data='x')]
    node_Ys = [float(y) for _, y in G.nodes(data='y')]

    # get north, south, east, west values either from bbox parameter or from the
    # spatial extent of the edges' geometries
    if shows is None:
        ### WARNING: EDITED
        edges = graph_to_gdfs_simple_graph(G, nodes=False, fill_edge_geometry=True)
        west, south, east, north = edges.total_bounds
        ### Edit region ended
    else:
        north, south, east, west = bbox

    # if caller did not pass in a fig_width, calculate it proportionately from
    # the fig_height and bounding box aspect ratio
    bbox_aspect_ratio = (north - south) / (east - west)
    if fig_width is None:
        fig_width = fig_height / bbox_aspect_ratio

    # create the figure and axis
    fig, ax = plt.subplots(figsize=(fig_width, fig_height), facecolor=bgcolor)
    ax.set_facecolor(bgcolor)

    # draw the edges as lines from node to node
    start_time = time.time()
    lines = []
    ### Notice: EDITED
    for u, v, data in G.edges(data=True):
        if 'geometry' in data and use_geom:
            # if it has a geometry attribute (a list of line segments), add them
            # to the list of lines to plot
            xs, ys = data['geometry'].xy
            lines.append(list(zip(xs, ys)))
        else:
            # if it doesn't have a geometry attribute, the edge is a straight
            # line from node to node
            x1 = G.nodes[u]['x']
            y1 = G.nodes[u]['y']
            x2 = G.nodes[v]['x']
```

```
            y2 = G.nodes[v]['y']
            line = [(x1, y1), (x2, y2)]
            lines.append(line)

    # add the lines to the axis as a linecollection
    lc = mpl.collections.LineCollection(lines, colors=edge_color, linewidths=edge_linewidth,
alpha=edge_alpha, zorder=2)
    ax.add_collection(lc)
    ox.log('Drew the graph edges in {:,.2f} seconds'.format(time.time() - start_time))

    # scatter plot the nodes
    ax.scatter(node_Xs, node_Ys, s=node_size, c=node_color, alpha=node_alpha,
edgecolor=node_edgecolor,
               zorder=node_zorder)

    # set the extent of the figure
    margin_ns = (north - south) * margin
    margin_ew = (east - west) * margin
    ax.set_ylim((south - margin_ns, north + margin_ns))
    ax.set_xlim((west - margin_ew, east + margin_ew))

    # configure axis appearance
    xaxis = ax.get_xaxis()
    yaxis = ax.get_yaxis()

    xaxis.get_major_formatter().set_useOffset(False)
    yaxis.get_major_formatter().set_useOffset(False)

    # if axis_off, turn off the axis display set the margins to zero and point
    # the ticks in so there's no space around the plot
    if axis_off:
        ax.axis('off')
        ax.margins(0)
        ax.tick_params(which='both', direction='in')
        xaxis.set_visible(False)
        yaxis.set_visible(False)
        fig.canvas.draw()

    if equal_aspect:
        # make everything square
        ax.set_aspect('equal')
        fig.canvas.draw()
    else:
        # if the graph is not projected, conform the aspect ratio to not stretch the plot
        if G.graph['crs'] == ox.settings.default_crs:
            coslat = np.cos((min(node_Ys) + max(node_Ys)) / 2. / 180. * np.pi)
            ax.set_aspect(1. / coslat)
            fig.canvas.draw()

    # annotate the axis with node IDs if annotate=True
    if annotate:
        for node, data in G.nodes(data=True):
            ax.annotate(node, xy=(data['x'], data['y']))

    # save and show the figure as specified
    fig, ax = ox.save_and_show(fig, ax, save, show, close, filename, file_format, dpi,
axis_off)
    return fig, ax


def graph_to_gdfs_simple_graph(G, nodes=True, edges=True, node_geometry=True, fill_edge_geom-
etry=True):
    """
```

```
    Convert a graph into node and/or edge GeoDataFrames
    Parameters
    ----------
    G : networkx multidigraph
    nodes : bool
        if True, convert graph nodes to a GeoDataFrame and return it
    edges : bool
        if True, convert graph edges to a GeoDataFrame and return it
    node_geometry : bool
        if True, create a geometry column from node x and y data
    fill_edge_geometry : bool
        if True, fill in missing edge geometry fields using origin and
        destination nodes
    Returns
    -------
    GeoDataFrame or tuple
        gdf_nodes or gdf_edges or both as a tuple
    """

    if not (nodes or edges):
        raise ValueError('You must request nodes or edges, or both.')

    to_return = []

    if nodes:

        start_time = time.time()

        nodes, data = zip(*G.nodes(data=True))
        gdf_nodes = gpd.GeoDataFrame(list(data), index=nodes)
        if node_geometry:
            gdf_nodes['geometry'] = gdf_nodes.apply(lambda row: Point(row['x'], row['y']),
axis=1)
        gdf_nodes.crs = G.graph['crs']
        gdf_nodes.gdf_name = '{}_nodes'.format(G.graph['name'])

        to_return.append(gdf_nodes)
        ox.log('Created GeoDataFrame "{}" from graph in {:,.2f} seconds'.for-
mat(gdf_nodes.gdf_name,
                                                                        time.time() -
start_time))

    if edges:

        start_time = time.time()

        # create a list to hold our edges, then loop through each edge in the
        # graph
        edges = []
        for u, v, data in G.edges(data=True):

            # for each edge, add key and all attributes in data dict to the
            # edge_details
            ### NOTE:EDITED
            edge_details = {'u': u, 'v': v}  # , 'key':key}
            for attr_key in data:
                edge_details[attr_key] = data[attr_key]

            # if edge doesn't already have a geometry attribute, create one now
            # if fill_edge_geometry==True
            if 'geometry' not in data:
                if fill_edge_geometry:
                    point_u = Point((G.nodes[u]['x'], G.nodes[u]['y']))
```

```
                    point_v = Point((G.nodes[v]['x'], G.nodes[v]['y']))
                    edge_details['geometry'] = LineString([point_u, point_v])
                else:
                    edge_details['geometry'] = np.nan

            edges.append(edge_details)
            ### Edit region end
        # create a GeoDataFrame from the list of edges and set the CRS
        gdf_edges = gpd.GeoDataFrame(edges)
        gdf_edges.crs = G.graph['crs']
        gdf_edges.gdf_name = '{}_edges'.format(G.graph['name'])

        to_return.append(gdf_edges)
        ox.log('Created GeoDataFrame "{}" from graph in {:,.2f} seconds'.for-
mat(gdf_edges.gdf_name,
                                                                        time.time() -
start_time))

    if len(to_return) > 1:
        return tuple(to_return)
    else:
        return to_return[0]


# Now override the function
ox.plot_graph = plot_graph_simple_graph
ox.graph_to_gdfs = graph_to_gdfs_simple_graph
ox.simplify_graph = simplify_simple_graph

# Plot the OSM graph now for preview
graph_osm_copy = graph_osmnx.copy()
graph_osm_copy.remove_nodes_from(list(nx.isolates(graph_osm_copy)))
nc = ['b' for node in graph_osmnx.nodes]
fig, ax = ox.plot_graph(graph_osmnx.to_undirected(), node_color=nc, node_zorder=2,
                        fig_height = 8, fig_width = 8, show=False, close=False)
legends = [mpl.lines.Line2D([],[], color='b', marker='.', linestyle='None', markersize=8, la-
bel='Nodes'),
           mpl.lines.Line2D([0],[0], color='grey', lw=1, label='OSM Pavements')]
ax.legend(handles=legends, loc='lower right')

# Plot the reference graph now for preview
graph_ref = read_shp_with_node([node_ref, edge_ref], simplify=False)
graph_ref_copy = graph_ref.copy()
graph_ref_copy.graph['name'] = 'Pedestrian graph at East Tsim Sha Tsui'
graph_ref_copy.graph['crs'] = ref_proj
graph_ref_copy = ox.simplify_graph(graph_ref_copy, True)
graph_ref_copy.remove_nodes_from(list(nx.isolates(graph_ref_copy)))
# Dummy name
nc = ['r' for node in graph_ref_copy.nodes]
fig, ax = ox.plot_graph(graph_ref_copy.to_undirected(), node_color=nc, node_edgecolor='k',
node_zorder=2,
                        fig_height = 8, fig_width = 8, equal_aspect=True, show=False,
close=False)
legends = [mpl.lines.Line2D([],[], color='r', marker='.', linestyle='None', markersize=8, la-
bel='Nodes'),
           mpl.lines.Line2D([0],[0], color='k', lw=1, label='Actual Pavements')]
ax.legend(handles=legends, loc='lower right')


# Now calculate selected metrics other than centrality

graph_osm_dir = graph_osm_copy.to_directed()
graph_ref_dir = graph_ref_copy.to_directed()
```

```
graph_osm_analysis = graph_osm_copy.to_undirected()
graph_ref_analysis = graph_ref_copy.to_undirected()

graph_osm_simple = nx.Graph(graph_osm_copy)
graph_ref_simple = nx.Graph(graph_ref_copy)

# Average Degree Connectivity
print("Average Degree Connectivity for OSM: "+str(nx.average_degree_connectiv-
ity(graph_osm_simple)))
print("Average Degree Connectivity for REF: "+str(nx.average_degree_connectiv-
ity(graph_ref_simple)))

# Average Path Length
print("Average Path Length for OSM: "+str(max([nx.average_shortest_path_length(C) for C in
nx.connected_component_subgraphs(graph_osm_simple)])))
print("Average Path Length for REF: "+str(max([nx.average_shortest_path_length(C) for C in
nx.connected_component_subgraphs(graph_ref_simple)])))

# Clustering Coefficient
print("Clustering Coefficient for OSM: " + str(nx.transitivity(graph_osm_simple)))
print("Clustering Coefficient for REF: " + str(nx.transitivity(graph_ref_simple)))

# Efficiency
print("Efficiency for OSM: " + str(nx.global_efficiency(graph_osm_analysis)))
print("Efficiency for REF: " + str(nx.global_efficiency(graph_ref_analysis)))

# Hierarchy
print("Flow hierarchy for OSM: " + str(nx.flow_hierarchy(graph_osm_dir)))
print("Flow hierarchy for REF: " + str(nx.flow_hierarchy(graph_ref_dir)))

# Now calculate centrality

# Degree Centrality
deg_centrality_osm = nx.degree_centrality(graph_osm_analysis)
deg_centrality_ref = nx.degree_centrality(graph_ref_analysis)

# Betweenness Centrality
btwn_centrality_osm = nx.betweenness_centrality(graph_osm_analysis)
btwn_centrality_ref = nx.betweenness_centrality(graph_ref_analysis)

# Closeness Centrality
close_centrality_osm = nx.closeness_centrality(graph_osm_analysis)
close_centrality_ref = nx.closeness_centrality(graph_ref_analysis)

# ECDF for centrality. Use for comparison
deg_centrality_osm_ecdf = ECDF([v for k, v in deg_centrality_osm.items()])
btwn_centrality_osm_ecdf = ECDF([v for k, v in btwn_centrality_osm.items()])
close_centrality_osm_ecdf = ECDF([v for k, v in close_centrality_osm.items()])

deg_centrality_ref_ecdf = ECDF([v for k, v in deg_centrality_ref.items()])
btwn_centrality_ref_ecdf = ECDF([v for k, v in btwn_centrality_ref.items()])
close_centrality_ref_ecdf = ECDF([v for k, v in close_centrality_ref.items()])

# Summarize statistics
centrality_osm = dict()
centrality_percentile_osm = dict()
for k, v in deg_centrality_osm.items():
    centrality_osm[k] = v
    centrality_percentile_osm[k] = deg_centrality_osm_ecdf(v)
for k, v in btwn_centrality_osm.items():
    if (centrality_osm[k] is not None):
        centrality_osm[k] = centrality_osm.get(k) + v
```

```
        centrality_percentile_osm[k] = centrality_percentile_osm.get(k) + btwn_central-
ity_osm_ecdf(v)
    else:
        centrality_osm[k] = v
        centrality_percentile_osm[k] = btwn_centrality_osm_ecdf(k)
for k, v in close_centrality_osm.items():
    if (centrality_osm[k] is not None):
        centrality_osm[k] = centrality_osm.get(k) + v
        centrality_percentile_osm[k] = centrality_percentile_osm.get(k) + close_central-
ity_osm_ecdf(v)
    else:
        centrality_osm[k] = v
        centrality_percentile_osm[k] = close_centrality_osm_ecdf(v)

centrality_ref = dict()
centrality_percentile_ref = dict()
for k, v in deg_centrality_ref.items():
    centrality_ref[k] = v
    centrality_percentile_ref[k] = deg_centrality_ref_ecdf(v)
for k, v in btwn_centrality_ref.items():
    if (centrality_ref[k] is not None):
        centrality_ref[k] = centrality_ref.get(k) + v
        centrality_percentile_ref[k] = centrality_percentile_ref.get(k) + btwn_central-
ity_ref_ecdf(v)
    else:
        centrality_ref[k] = v
        centrality_percentile_ref[k] = btwn_centrality_osm_ecdf(v)
for k, v in close_centrality_ref.items():
    if (centrality_ref[k] is not None):
        centrality_ref[k] = centrality_ref.get(k) + v
        centrality_percentile_ref[k] = centrality_percentile_ref.get(k) + close_central-
ity_ref_ecdf(v)
    else:
        centrality_ref[k] = v
        centrality_percentile_ref[k] = close_centrality_osm_ecdf(v)

# Prepare to show the result
l_centrality_osm = [v / 3 for k, v in centrality_osm.items()]
l_centrality_osm_ecdf = ECDF(l_centrality_osm)
l_centrality_osm_sum_ecdf = [l_centrality_osm_ecdf(v) for v in l_centrality_osm]
l_centrality_percentile_osm = [v / 3 for k, v in centrality_percentile_osm.items()]

l_centrality_osm.sort(reverse=True)
l_centrality_osm_high = l_centrality_osm[:len(l_centrality_osm) // 10]

l_centrality_percentile_osm = np.array(l_centrality_percentile_osm)
l_centrality_percentile_osm[::-1].sort()

l_centrality_ref = [v / 3 for k, v in centrality_ref.items()]
l_centrality_ref_ecdf = ECDF(l_centrality_ref)
l_centrality_percentile_ref = [v / 3 for k, v in centrality_percentile_ref.items()]

l_centrality_ref.sort(reverse=True)
l_centrality_ref_high = l_centrality_ref[:len(l_centrality_ref) // 10]

l_centrality_percentile_ref = np.array(l_centrality_percentile_ref)
l_centrality_percentile_ref[::-1].sort()

# Figure: sum of three centralities
plt.plot(l_centrality_osm, label='OSM')
plt.plot(l_centrality_ref, label='Actual')

plt.legend(loc='upper right')
```

```
plt.grid(True, color=(0.85,0.85,0.85))
plt.title("Sum of the Three Centralities Measures")
plt.xlabel("n-th most central nodes")
plt.ylabel("Sum of the centralities")

#CDF of summing three centralities

plt.hist(l_centrality_osm, normed=True, cumulative=True, label='OSM',
         histtype='stepfilled', color=(0,0,1,0.75))
plt.hist(l_centrality_ref, normed=True, cumulative=True, label='Actual',
         histtype='stepfilled', color=(1,0,0,0.4))
#legends = [mpl.lines.Line2D([],[], color='b' label='OSM'), mpl.lines.Line2D([],[], color='r'
label='Actual')]
plt.legend(loc='upper left')
plt.title("Cumulative Distribution of Centralities Measures")
plt.xlabel("Sum of the centralities")
plt.ylabel("Probability")

# Use Percentile instead of sum of three centralities values

plt.plot(l_centrality_percentile_osm, label="OSM")
plt.plot(l_centrality_percentile_ref, label="Actual")

plt.legend(loc='upper right')
plt.grid(True, color=(0.85,0.85,0.85))
plt.title("Centralities Percentile")
plt.xlabel("n-th most central nodes")
plt.ylabel("Percentile of the sum of three centralities")

# Plot the Graph to show central nodes

central_nc_osm = []
for node in graph_osm_analysis.nodes():
    if (centrality_osm.get(node)/3) in l_centrality_osm_high:
        central_nc_osm.append('orange')
    else:
        central_nc_osm.append('b')
fig, ax = ox.plot_graph(graph_osm_analysis, node_color=central_nc_osm, node_edgecolor='grey',
node_zorder=2,
                        fig_height = 10, fig_width = 10, equal_aspect=True, show=False,
close=False)
legends = [mpl.lines.Line2D([],[], color='b', marker='.', linestyle='None', markersize=8, la-
bel='Nodes'),
           mpl.lines.Line2D([],[], color='orange', marker='.', linestyle='None', mark-
ersize=8, label='Central nodes'),
           mpl.lines.Line2D([0],[0], color='grey', lw=1, label='OSM Pavements')]
ax.legend(handles=legends, loc='lower right')


central_nc_ref = []
for node in graph_ref_analysis.nodes():
    if (centrality_ref.get(node)/3) in l_centrality_ref_high:
        central_nc_ref.append('cyan')
    else:
        central_nc_ref.append('r')
fig, ax = ox.plot_graph(graph_ref_analysis, node_color=central_nc_ref, node_edgecolor='k',
node_zorder=2,
                        fig_height = 10, fig_width = 10, equal_aspect=True, show=False,
close=False)
legends = [mpl.lines.Line2D([],[], color='r', marker='.', linestyle='None', markersize=8, la-
bel='Nodes'),
           mpl.lines.Line2D([],[], color='cyan', marker='.', linestyle='None', markersize=8,
label='Central nodes'),
```

```
            mpl.lines.Line2D([0],[0], color='k', lw=1, label='Actual Pavements')]
ax.legend(handles=legends, loc='lower right')
```