MSC PROJECT INVESTIGATION REPORT

DATA-DRIVEN MODELING OF MOBILITY IN SCOTLAND

AFOLABI SAMAD OLATOYE
(1910611)

A REPORT SUBMITTED AS PART OF THE REQUIREMENTS FOR THE DEGREE OF MSC IN IT WITH CYBERSECURITY AT ROBERT GORDON UNIVERSITY, ABERDEEN, SCOTLAND.

FEBRUARY, 2021

SUPERVISOR: KATE HAN

DECLARATION

I confirm that the work contained in this document has been composed solely by myself and has not been accepted in any previous application for a degree.
All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signature: ……………………………………………..

Date: …………………….

ACKNOWLEDGEMENT

I would like to thank my supervisor Kate Han along with Lee Christie and Ciprian Zavoianu for their patience and assistance in the development of this project.
My appreciation goes to friends and family for their support during the course of this project development.

ABSTRACT

Passenger transport networks are one of the most used modes of transport in Scotland. They are cheap and convenient for the people within commuting distance of their jobs or homes. It is not far fetched to assume that analysis of transport data would produce valuable insights. The passenger transport data can help in urban planning, disaster planning, marketing and other location based services. The technological improvements over the years has also yielded better tools to analyze transport data and present a better understanding of the geographical area. The usage of passenger transport networks also leads to a massive amount of data being generated by transport operators. There is a demand for more ways to analyse this data. One issue this project tackles the development of a tool to extend the capability of a software used for analysing the data generated. Additionally, the project designs and implements a means for analysing reachability using the data generated.

TABLE OF CONTENTS

1 INTRODUCTION

Mobility is a multidisciplinary field of physics and computer science. Models and predictive approaches have been developed to understand and improve mobility. However, the increase in the use of multisource open data has proved to be a driving force for exploring mobility patterns from a quantitative and microscopic perspective. The studies of mobility through data-driven modeling has a vital role in applications such as urban planning, intelligent public transportation management and location-based services (Wang et al 2019).

Passenger transport refers to the total movement of passengers using inland transport on a given network (OECD 2021). According to transport statistics published by Transport Scotland, over 502 million public transport journeys were made by air(6%), bus(73%), rail(19%) and ferry(2%) in 2019/2020. There are fewer vehicles in Scotland than in the United Kingdom as a whole which suggests people depend on the public transport network in Scotland. The data highlights the importance of passenger transport networks in Scotland. Improvements in the passenger transport network will probably lead to an increase in its usage. This lends credence to developing models to understand, analyse and visualise the transport network (Transport Scotland 2020).

Also, there are data sources available to perform this analysis. Tools such as OpenStreetMap which is a collection of knowledge that provides user-generated street maps. The process of mapping the earth was until recent times done by highly skilled, well trained and equipped groups or individuals. The development of OpenStreetMap can be attributed to technological changes, increased bandwidth and crowdsourcing. Crowdsourcing is how a congregation of users can perform functions that are either difficult or expensive to implement. In this instance, the function is providing “map data that is free to use, editable and licensed under new copyright schemes” (Mordecai et al 2020).

Additionally, transport network operators generate timetable data which is stored in different formats across proprietary databases. Formats used in the transport industry include ATCO-CIF, TransXChange and General Transit Feed Specification (GTFS). These formats allow transport network operators to share timetable data which provide information on journey start and stop times, routes and date of operation.

1.1 MOTIVATION

Robert Gordon University's Smart Data Technologies Centre (SDTC) research team and Highlands and Islands Transport Partnership (HITRANS) have a partnership which lead to the development of an interactive computerised model to visualise and analyse the Scottish passenger transport network. The result of this collaboration is a fully functioning software developed using the Java programming language for performing inbound and outbound reachability analysis using OpenStreetMap and GTFS data.

Transport data is currently available in different formats therefore there is a need for an extension to the current software to support ATCO-CIF which is considered a legacy format and an implementation of the A* (A-star) algorithm using the Java programming language to aid in reachability analysis.

1.2 REPORT STRUCTURE

Chapter 2 presents the background research that was conducted. It consists of two major sub-chapters: Transport data formats and Graph Algorithms. In the transport data section, data formats used to store transport data is discussed. In the Graph Traversal Algorithms section, the graph data structures and algorithms is discussed as well as the reason behind the use of the data structure and algorithm in modeling transport networks.

Chapter 3 provides specification on how the extension would be developed to achieve the project aims and objectives.

Chapter 4 provides a conclusion on the proposed software solution and background research carried out prior to implementation is discussed.

2 RELATED WORKS

2.1 TRANSPORT DATA FORMATS

The passenger transport network has seen an increase in usage as people prefer to commute from their jobs to their homes. The addition of new services to the network has been crucial in providing comfortable and faster travel facilities. A look at the transport network of a geographical area provides a glimpse into the lifestyle of the area. The data of transport networks have been made available for the public recently, this combined with the variety of tools available today can allow for interesting discoveries (Prommaharaj et al 2020).

However, analysing such data can be challenging because of its multidimensional nature and the use of different formats by transport network operators. An example is a route that consists of a couple of journeys a service takes and each journey passes through many stops with the stops being a part of many routes. Data formats used by network operators mostly contain the same information such as routes, stops, trips and others.

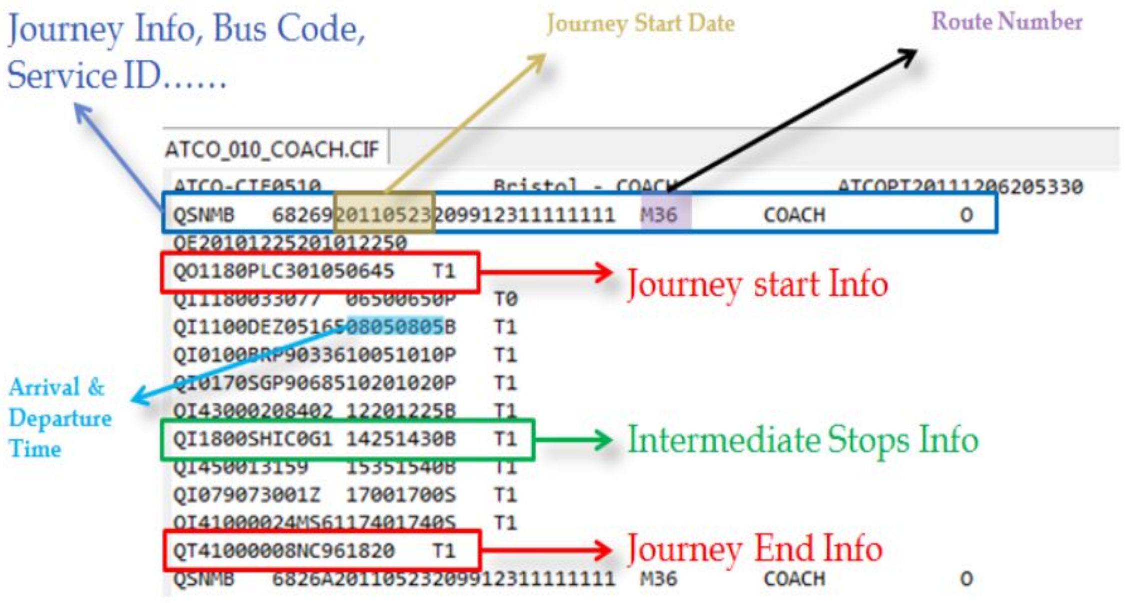
The two transport data formats in relation to this project are discussed further. The ATCO-CIF and the GTFS. The ATCO-CIF is the format this project intends to add functionality for and the GTFS is the current format the software supports. Below is a list of definitions associated with transport timetable data:

1) Journey: The movement of a vehicle (bus, train, ferry or coach) over a sequence of stopping points and the times from its origin terminal point to a destination terminal point.
2) Service: A mode of public transport which can be either a bus, train, ferry or coach.
3) Stops: The geographical location at which an event happens during a course of a Journey.
4) Route: A group of Journeys with more or less identical stopping points.
5) Route number: A unique identifier of each route.
6) Events: Describes what happens to a vehicle at each stop or timing point during the course of a journey.
7) Clusters: Geographical groupings of stops at which it is possible to change journeys.
8) Interchange time: Minimum time needed to change between journeys at a stop or within a cluster.

2.1.2 ATCO CIF SPECIFICATION

The Association of Transport Coordinating Officers Common Interface File (ATCO-CIF) specification defines a format used for the interchange of timetable data between public transport agencies in the UK. Files that use this specification end in ".cif". Public transport agencies in the UK make use of file formats defined by atco-cif to share timetable data.

The specification defines terms associated with the transportation industry. These terms are defined in Appendix A of this report. The timetable data stored using the CIF file format contains information on the journey, stop locations and schedules of the transport network. The file format defines a record per row. Each record starts with two characters which act as identifiers for the record type (Traveline 2006).



```
ATCO-CIF0510                Bristol - COACH                ATCOPT20111206205330
QSNMB    6826920110523209912311111111  M36       COACH            O
QE2010122520101225O
QO1180PLC301050645     T1
QI1180033077   06500650P    T0
QI1100DEZ05165080508O5B     T1
QI0100BRP9033610051010P     T1
QI0170SGP9068510201020P     T1
QI43000208402  12201225B    T1
QI1800SHIC0G1  14251430B    T1
QI450013159    15351540B    T1
QI079073001Z   17001700S    T1
QI41000024MS6117401740S     T1
QT41000008NC961820     T1
QSNMB    6826A20110523209912311111111  M36       COACH            O
```

Fig 1 - ATCO-CIF Example FIle format [12]

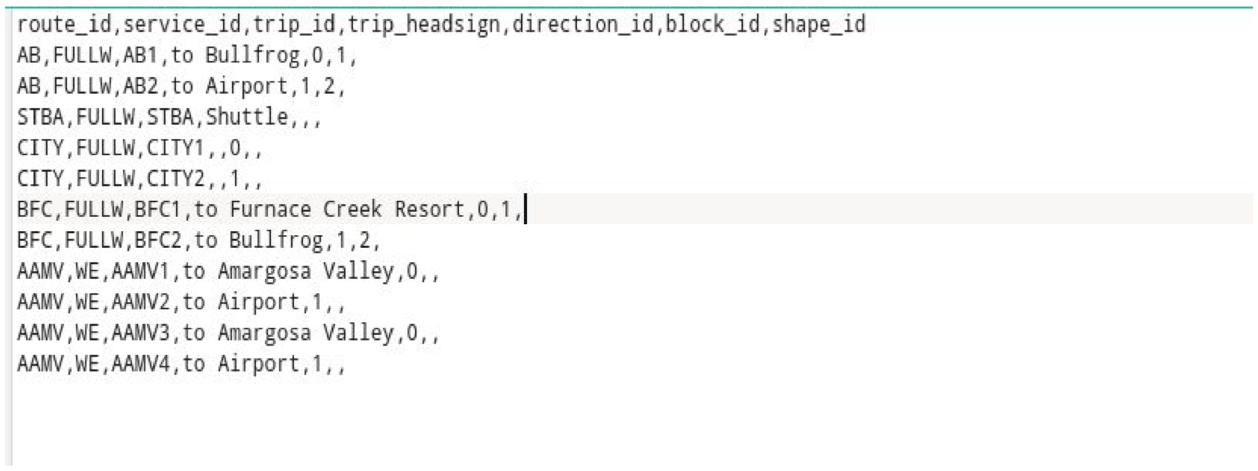
The first row is reserved for the file header information. Subsequent rows contain records about the timetable data. Modeling using this data will provide a near real time view of the public transport system. The CIF file format is still widely used in the UK by transport agencies such as traveline for planning journeys on public transport networks in the UK.

Agencies such as traveline develop tools in-house for ingesting and interpreting CIF file formats. The software developed by SDTC does not currently possess the capability to read and interpret the CIF file format. Providing that capability as part of the extension is one of the objectives of this project.

2.1.3 GTFS

The General Transit Feed Specification is a data format used to describe information about public transport systems similar to ATCO-CIF. Developed by Google along with TriMet to tackle the issue of online transit planners. GTFS allows transport agencies to share transit information in an open source format which has encouraged agencies to share data to the public.

GTFS presents transport data in a collection of comma-delimited text files compressed into a ZIP file. Files in a GTFS feed are CSV files but use an extension of ".txt". Files required as part of the GTFS specification can have additional columns which leads to different levels of detail among feeds from different operators (Quentin 2014).



```
route_id,service_id,trip_id,trip_headsign,direction_id,block_id,shape_id
AB,FULLW,AB1,to Bullfrog,0,1,
AB,FULLW,AB2,to Airport,1,2,
STBA,FULLW,STBA,Shuttle,,,
CITY,FULLW,CITY1,,0,,
CITY,FULLW,CITY2,,1,,
BFC,FULLW,BFC1,to Furnace Creek Resort,0,1,
BFC,FULLW,BFC2,to Bullfrog,1,2,
AAMV,WE,AAMV1,to Amargosa Valley,0,,
AAMV,WE,AAMV2,to Airport,1,,
AAMV,WE,AAMV3,to Amargosa Valley,0,,
AAMV,WE,AAMV4,to Airport,1,,
```

Fig 2 - Sample Feed of routes in a GTFS dataset

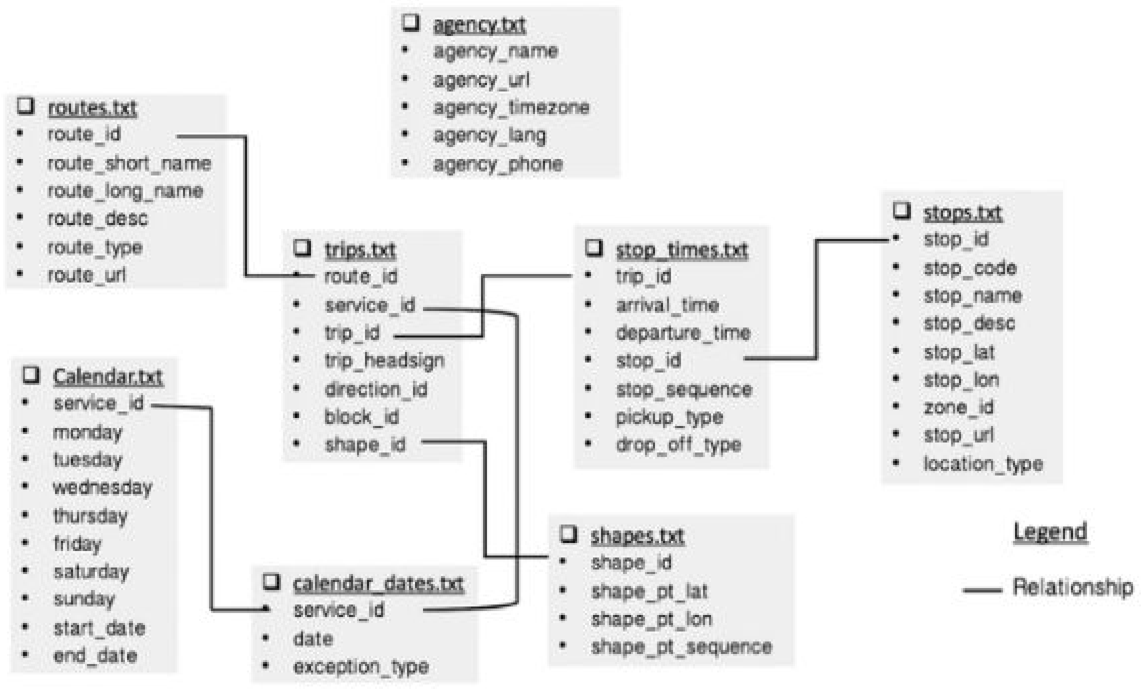


Fig 3 - Relationship between files in a GTFS Feed [13]

Each GTFS zip file makes up a complete feed and any changes made to a text file in this feed creates a new feed. These files have a relationship similar to tables in a relational database. GTFS defines files to be included in a feed. These file definitions are outlined in Appendix B of this project report. The current model developed by SDTC makes use of the GTFS format for analysing transport data.

2.3 GRAPH TRAVERSAL ALGORITHMS

Graphs are a representation of objects where some pairs of objects are connected by links. The interconnected objects are represented by points called vertices and the links that connect these vertices are called edges. Formally, a graph is a pair of sets **V**, **E** where **V** is the set of vertices and **E** is a set of edges. In the figure below, each node represents a vertex and the lines connecting them represent edges.

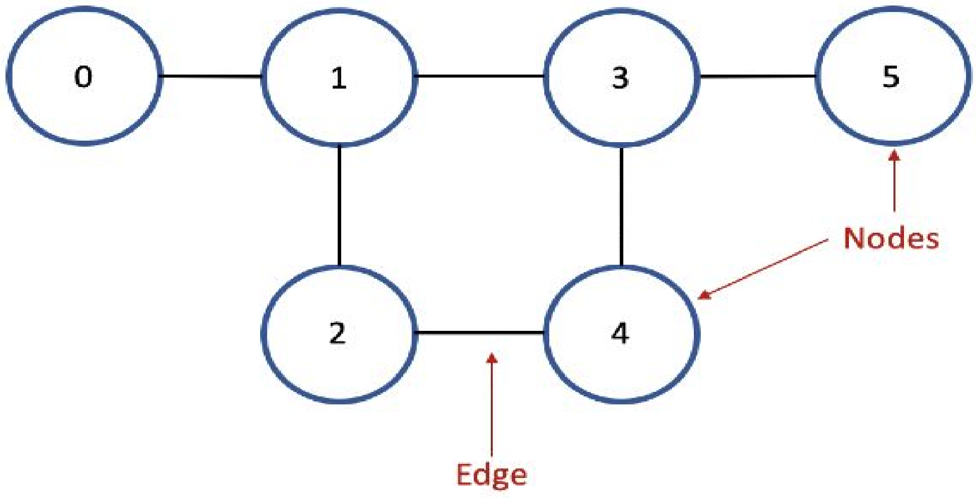


Fig 4 - Example of a Graph [23]

There are different types of graphs which include:

- Undirected graphs which have directionless edges between vertices.
- Directed graphs or digraphs which have edges that point in a particular direction. This means edges can only be traversed in the particular direction the arrow points to.
- Weighted graphs have edges with non-zero values which can be positive or negative.
- Unweighted graphs have edges with zero-values.

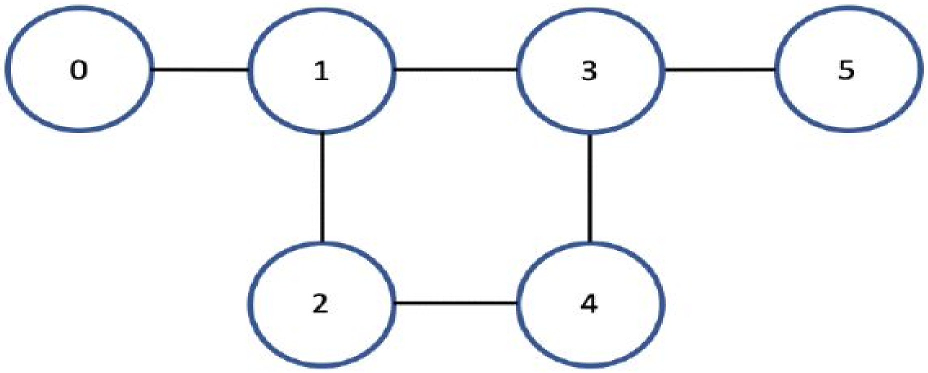


Fig 5 - Unweighted and Undirected graph Example [23]

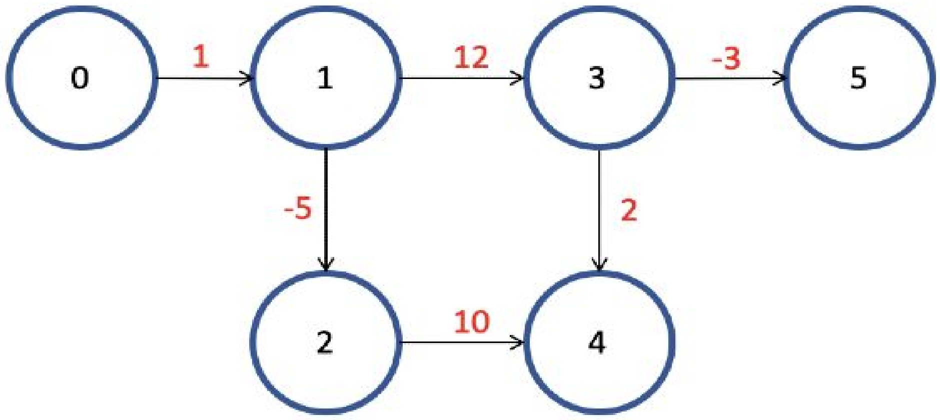


Fig 6 - Weighted and Directed graph Example [23]

Traversal is the movement through a structure visiting each of the vertices once. There are two possible methods to traverse a graph: Breadth-first and Depth-First. The two methods are discussed further.

1. Breadth-first: This method visits all the vertices, beginning with a specified start vertex. Vertices are only visited once and they are only visited if there is a path from the start vertex. It makes use of a queue data structure which holds a list of vertices to be visited soon. Since a queue is a first in first out structure, vertices are visited in the order in which they are added. Vertices are not added to the queue if they are already there or have been visited.
2. Depth-first: It works the same way as breadth-first except neighbors of vertices are added to a stack data structure. Since a stack data structure is a last in first out structure, vertices are visited in the order in which they are popped from the stack.

Transport networks are represented using weighted graphs. One of the problems encountered in transport networks is determining the shortest path between two vertices on the graph. Algorithms are recipes for solving a problem logically. One of the problems that graph traversal algorithms tackle is the shortest path search problem essential to route planning for the interactive model developed by SDTC. Algorithms used to tackle this problem include: Dijkstra and A*. The path search problem involves finding the best path between a starting location and a destination under provided conditions. The two algorithms are discussed further.

2.3.1 DIJKSTRA ALGORITHM

Dijkstra algorithm is an algorithm used to compute the shortest path from a single source to each of the vertices in a graph. Dijkstra's algorithm uses the greedy approach to solve the single source shortest path problem. For the algorithm to work, the graph should be directed and the edges should be non-negative (Goyal et al 2014). The shortest path cannot be acquired if the edges are negative. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra‘s algorithm can be used to find the shortest route between one city and all other cities. Dijkstra algorithm uses the priority queue data structure which unlike the queue data structure removes items in order of value rather than order of being added.

The algorithm works by keeping the shortest distance of Vertex **V** from the source in an array, Dist1. The shortest distance of the source to itself is zero. Distance for all other vertices is set to infinity to indicate that those vertices have not yet been processed. After the algorithm finishes the processing of the vertices, Dist1 will have the shortest distance of vertex from source to every other vertex. Two sets are maintained which helps in the processing of the algorithm, in the first set all the vertices that have been processed are maintained i.e. for which the shortest path has been computed. And in the second set, all other vertices not yet processed are maintained (Goyal et al 2014).

One of the main uses of Dijkstra is in finding the shortest paths where the destination may not be known. An example is if you possess locations for several resources but the closest one is the question being posed. Dijkstra has an advantage here since it performs a blind search through these locations to determine shortest paths to each location.

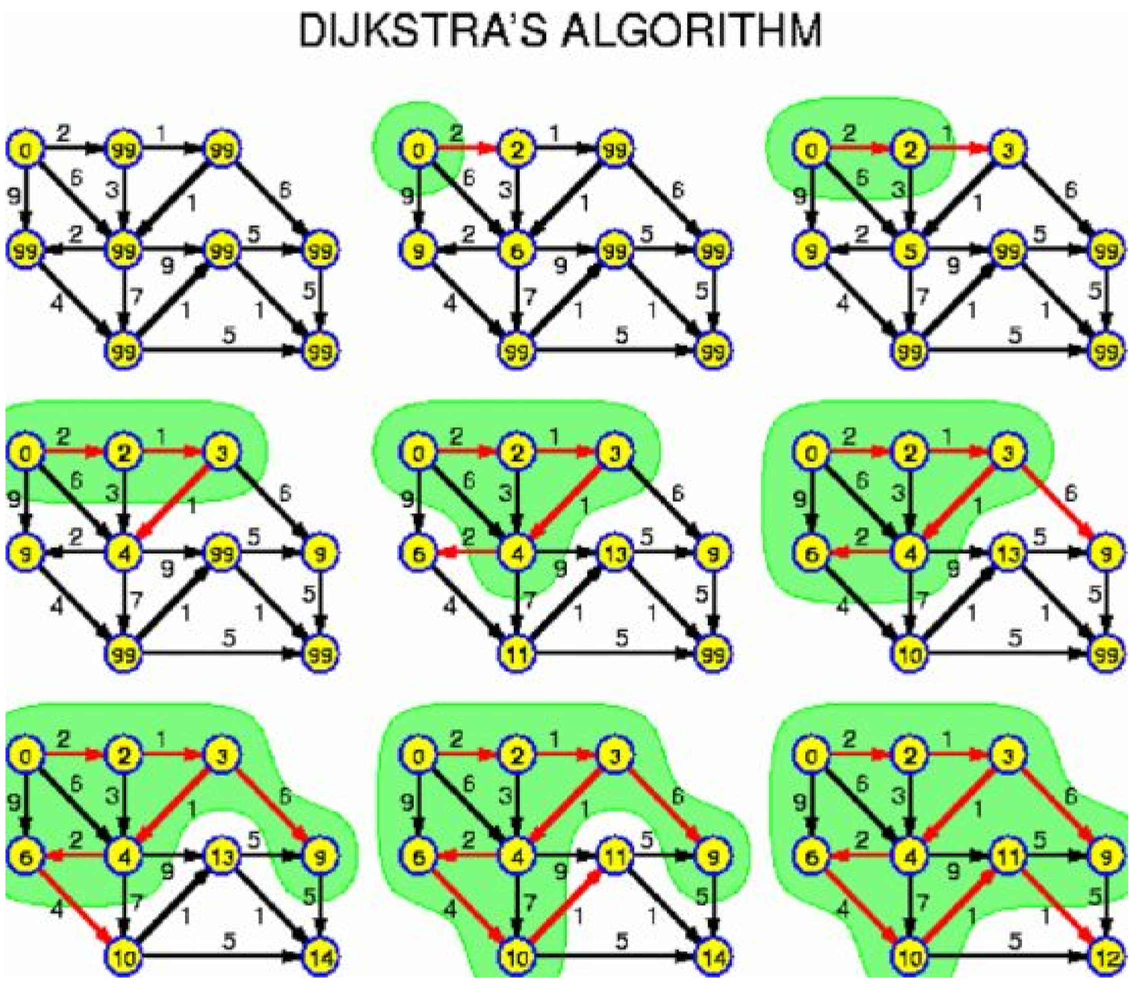


Fig 7 - Example of Dijkstra in steps [17]

2.3.2 A-STAR ALGORITHM

A-star algorithm is an algorithm that finds the path from a given initial node to a destination node. It uses a heuristic estimate h(n) that provides an estimate of the best route that goes through that node. In A* implementations, g(n) is a terminology which represents the exact cost of a path from the source to any vertex n and h(n) represents a heuristic estimate from vertex n to the destination or goal (Goyal et al 2014).

As indicated in A* based Pathfinding (Cui, X et al 2011), A* possesses some useful properties which include:

- It is guaranteed to find a path from the source to a destination if there exists a path.
- The path is optimal if h(n) is an admissible heuristic i.e. h(n) is less than or equal to the actual cost of the path from vertex n to the destination.
- It makes efficient use of the heuristic i.e. it examines fewer nodes to find optimal paths.

Heuristic is a technique that improves the efficiency of the search process with the possibility of sacrificing completeness. The use of an heuristic is one of the differences between A* and dijkstra. A* has an heuristic function of $f(n) = g(n) + h(n)$ while dijkstra has a heuristic function of $f(n) = g(n)$ which is the same as just $g(n)$. As described in Path Finding (Goyal et al 2014), the time it takes for A* to execute is faster than the time taken by dijkstra with no difference in the length of the path taken by the two algorithms. Also described by Path Finding (Goyal et al 2014), dijkstra and A* both get the same results but the time taken by each varies. A* uses the best first search approach while dijkstra uses greedy best first approach. This means dijkstra searches more parts of the graph leading to a waste of resources. A* searches the graph in the direction of the destination, this is due to the use of an heuristic.

3 PROJECT SPECIFICATION

This project is about designing and implementing a software extension that can process the CIF date format and implements the A* algorithm. The software extension should be able to read and convert cif files to gtfs feeds that can be processed by the current software implementation. Then the software extension will be running an implementation of A* algorithm and display results from the algorithm using the gtfs feeds generated as input.

3.1 PROPOSED SOFTWARE SOLUTION

3.1.1 FUNCTIONAL REQUIREMENTS

The functional requirements consist of the functionalities that are required and optional to meet the project objectives. However, there is no guarantee that the optional requirements would be implemented because of the limited time associated with the project. The functional requirements for the proposed software solution are listed below:

1) Provide a user with the capability of selecting files to be processed i.e. it allows a user to select and upload which CIF file is to be processed.
2) A button which allows the user to initiate the process of converting CIF files to GTFS feeds should be made available.
3) A dialog window to display the relevant information to the user.
4) A button which allows a user to select files to be used as input to display on maps.
5) A window providing a visual of transport networks using OpenStreetMap and the GTFS files generated.
6) A window for editing the implementation such as the starting terminal or node and the destination node or terminal.
7) A button to exit the software should also be provided.
8) Optionally provide a window for users to explore generated GTFS files.
9) Optionally provide a window that allows selecting location to store the generated GTFS files.

3.1.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements consist of functionalities to help improve the user experience in the process of operating the proposed software solution. The following list of non-functional requirements is not in any way exhaustive. Improvements would be made continuously during the course of developing the proposed software solution:

1) The Graphical User Interface (GUI) of the software should be user friendly. It should not be overstuffed with buttons and should have a simple interface.
2) The GUI should be self explanatory. A user should be able to make use of the software without needing specialised knowledge. Buttons and labels should be named properly.
3) The software should be small in size and not require too much computer resources to perform its necessary tasks.
4) The GUI should properly generate dialog windows to indicate errors.
5) The GUI should have a menu section to allow for easy navigation between functionalities.

3.1.3 PROPOSED IMPLEMENTATION

The proposed software solution will be developed using Apache Netbeans and the Java programming language. The reason behind the use of Java is that the codebase for the current SDTC software uses Java as its development and Netbeans is chosen as the IDE because it's free and open source. Netbeans also provides a tool for developing GUI interfaces without coding which would speed up development time.

4 CONCLUSION

In this project investigation, a software solution to act as an extension for a software developed by SDTC is proposed. This software will require users to select and upload CIF files, initiate the processing of the CIF file, define origin and destination to perform analysis on. Results are displayed for the user in a dialog window to indicate files have been processed or implementation of the A* algorithm has been rendered graphically.

The software would display the transport data generated using OpenStreetMap and the GTFS Feed generated after processing the CIF files. The proposed software has been developed for academic purposes and commercial use would be subject to any existing laws.

BIBLIOGRAPHY

[1] HITRANS 2011. Available from: https://hitrans.org.uk/Home [Accessed February 28 2021]
[2] OpenStreetMap Wiki contributors 2021. Available from: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap [Accessed February 18 2021]
[3] Google Developers 2020. Reference. Available from: https://developers.google.com/transit/gtfs/reference [Accessed February 28 2021]
[4] Traveline 2006. Timetables. Available from: https://web.archive.org/web/20060224065635/http://www.travelinedata.org.uk/times.htm#cif [Accessed February 28 2021]
[5] Traveline 2006. ATCO-CIF Specification. Available from: https://web.archive.org/web/20051115090734/http://www.travelinedata.org.uk/CIF/atco-cif-spec.pdf [accessed February 28 2021]
[6] Corsar, D., Edwards, P., Nelson, J., Baillie, C., Papangelis, K. and Velaga, N., 2017. Linking open data and the crowd for real-time passenger information. Journal of Web Semantics, 43, pp.18-24.
[7] Krumnow, D.I.M., Modeling Mobility with Open Data.
[8] Wang, J., Kong, X., Xia, F. and Sun, L., 2019. Urban human mobility: Data-driven modeling and prediction. ACM SIGKDD Explorations Newsletter, 21(1), pp.1-19.
[9] OECD 2021. Passenger transport (indicator). Available from: https://data.oecd.org/transport/passenger-transport.htm [Accessed February 28 2021]
[10] Transport Scotland 2020. Scottish Transport Statistics. Available from: https://www.transport.gov.scot/media/49177/scottish-transport-statistics-2020-publication-final-version.pdf [Accessed February 28 2021]
[11] Mordecai Haklay, Patrick Weber 2020. OpenStreetMap. Available from: https://discovery.ucl.ac.uk/id/eprint/13849/1/13849.pdf [Accessed February 25 2021]
[12] Berenjkoub, M., Nyshadham, H., Deng, Z. and Chen, G., 2015. A Visual Analytic System for Longitudinal Transportation Data of Great Britain.
[13] Prommaharaj, P., Phithakkitnukoon, S., Demissie, M.G., Kattan, L. and Ratti, C., 2020. Visualizing public transit system operation with GTFS data: A case study of Calgary, Canada. Heliyon, 6(4), p.e03729.

[14] Quentin Zervaas 2014. The Definitive Guide To GTFS. Available from: http://gtfsbook.com/gtfs-book-sample.pdf [Accessed March 1 2021]

[15] Jim Paterson GCU Module 2012. Graph Data Structures. Available from: http://www.paterson.co.uk/gcal/lecturenotes/graph_ds.pdf [Accessed March 1 2021]

[16] Tutorialpoint. Data Structure and Algorithms Graph data Structure. Available from: https://kaspercphbusiness.github.io/advprog/Lektion04-graphsAndBacktrack/Data%20Structures%20and%20Algorithms%20Graph%20Data%20Structure.pdf [Accessed March 1 2021]

[17] Jasika, N., Alispahic, N., Elma, A., Ilvana, K., Elma, L. and Nosovic, N., 2012, May. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In 2012 proceedings of the 35th international convention MIPRO (pp. 1811-1815). IEEE.

[18] Goyal, A., Mogha, P., Luthra, R. and Sangwan, N., 2014. Path finding: A* or Dijkstra's. International Journal in IT and Engineering, 2(1), pp.1-15.

[19] Cui, X. and Shi, H., 2011. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, *11*(1), pp.125-130.

[20] Ravikiran Jaliparthi Venkat 2014. Path Finding - Dijkstra's Algorithm. Available from: https://cs.indstate.edu/~rjaliparthive/dijkstras.pdf [Accessed February 28 2021]

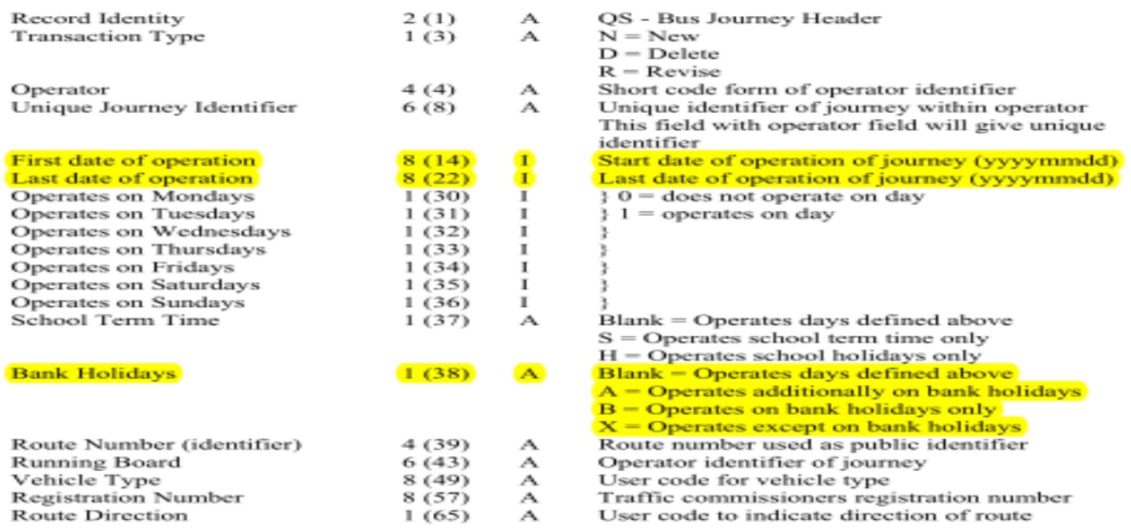
[21] Delling, D., Sanders, P., Schultes, D. and Wagner, D., 2009. Engineering route planning algorithms. In *Algorithmics of large and complex networks* (pp. 117-139). Springer, Berlin, Heidelberg.

[22] Zeng, W. and Church, R.L., 2009. Finding shortest paths on real road networks: the case for A. *International journal of geographical information science*, *23*(4), pp.531-543.

[23] Tenson Cai 2020. Graph Implementation and Traversal Algorithms. Available from: https://www.section.io/engineering-education/graph-traversals-java/ [Accessed March 1 2021]

APPENDIX A: ATCO-CIF SPECIFICATION

JOURNEY HEADER

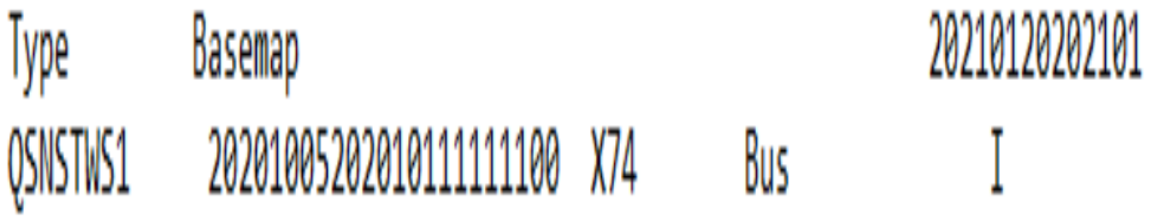


Record Identity	2 (1)	A	QS - Bus Journey Header

Transaction Type	1 (3)	A	N = New
			D = Delete
			R = Revise

Operator	4 (4)	A	Short code form of operator identifier
Unique Journey Identifier	6 (8)	A	Unique identifier of journey within operator This field with operator field will give unique identifier
First date of operation	8 (14)	I	Start date of operation of journey (yyyymmdd)
Last date of operation	8 (22)	I	Last date of operation of journey (yyyymmdd)
Operates on Mondays	1 (30)	I	} 0 = does not operate on day
Operates on Tuesdays	1 (31)	I	} 1 = operates on day
Operates on Wednesdays	1 (32)	I	}
Operates on Thursdays	1 (33)	I	}
Operates on Fridays	1 (34)	I	}
Operates on Saturdays	1 (35)	I	}
Operates on Sundays	1 (36)	I	}
School Term Time	1 (37)	A	Blank = Operates days defined above S = Operates school term time only H = Operates school holidays only
Bank Holidays	1 (38)	A	Blank = Operates days defined above A = Operates additionally on bank holidays B = Operates on bank holidays only X = Operates except on bank holidays
Route Number (identifier)	4 (39)	A	Route number used as public identifier
Running Board	6 (43)	A	Operator identifier of journey
Vehicle Type	8 (49)	A	User code for vehicle type
Registration Number	8 (57)	A	Traffic commissioners registration number
Route Direction	1 (65)	A	User code to indicate direction of route

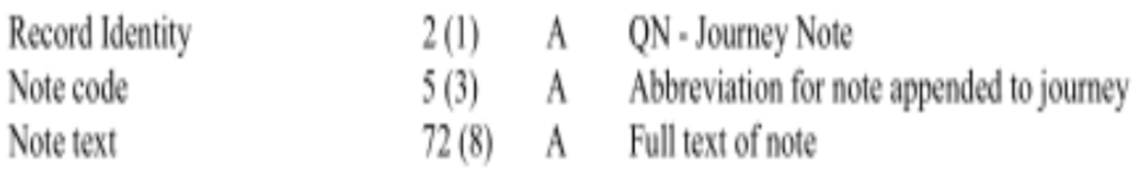
Fig 8 - Journey Header Specification [5]



```
Type      Basemap                                  20210120202101
QSNSTWS1   2020100520201011111100  X74     Bus          I
```

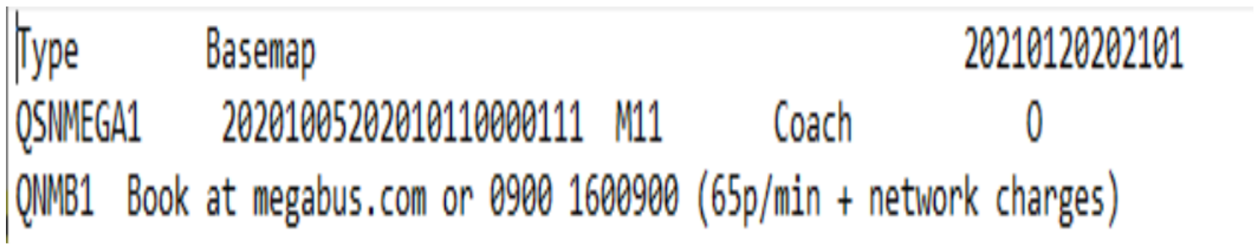
Fig 9 - Journey Header Example [5]

JOURNEY NOTE RECORD



Record Identity	2 (1)	A	QN - Journey Note
Note code	5 (3)	A	Abbreviation for note appended to journey
Note text	72 (8)	A	Full text of note

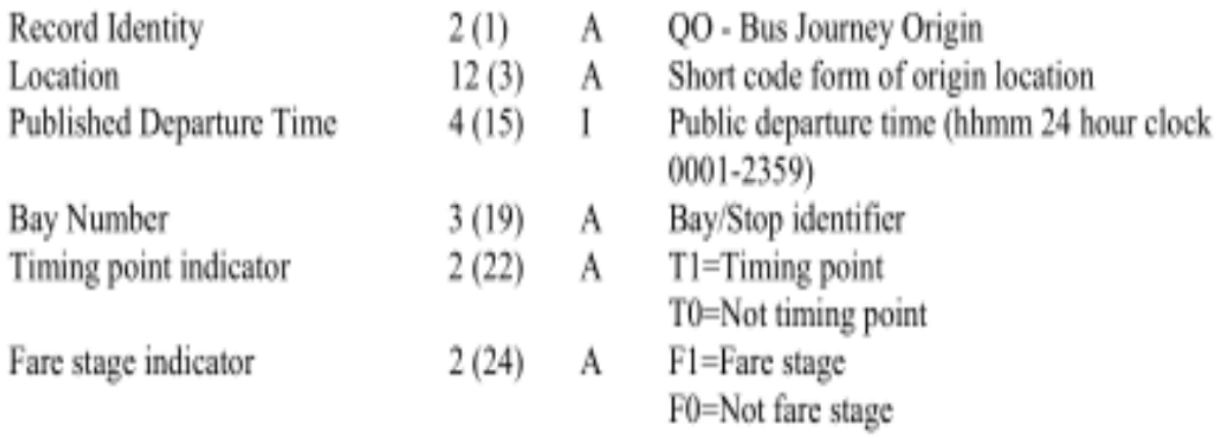
Fig 10 - Journey Note Record Specification [5]



```
Type        Basemap                                        20210120202101
QSNMEGA1    20201005202010110000111  M11       Coach          0
QNMB1  Book at megabus.com or 0900 1600900 (65p/min + network charges)
```

Fig 11 - Journey Note Record Example [5]

ORIGIN RECORD

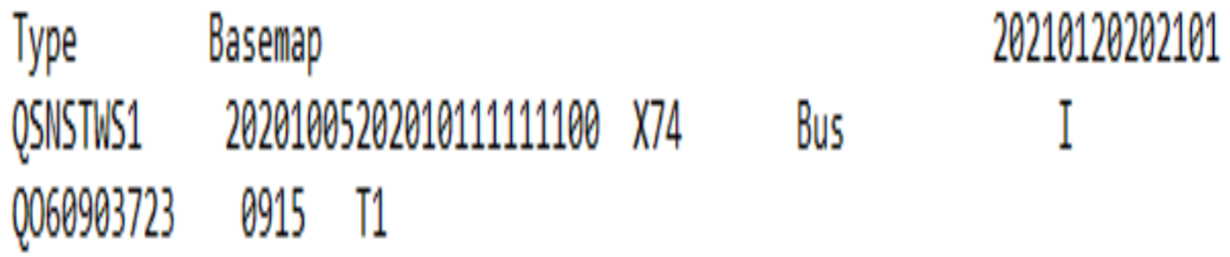


Record Identity	2 (1)	A	QO - Bus Journey Origin
Location	12 (3)	A	Short code form of origin location
Published Departure Time	4 (15)	I	Public departure time (hhmm 24 hour clock 0001-2359)
Bay Number	3 (19)	A	Bay/Stop identifier

Timing point indicator	2 (22)	A	T1=Timing point
			T0=Not timing point

Fare stage indicator	2 (24)	A	F1=Fare stage
			F0=Not fare stage

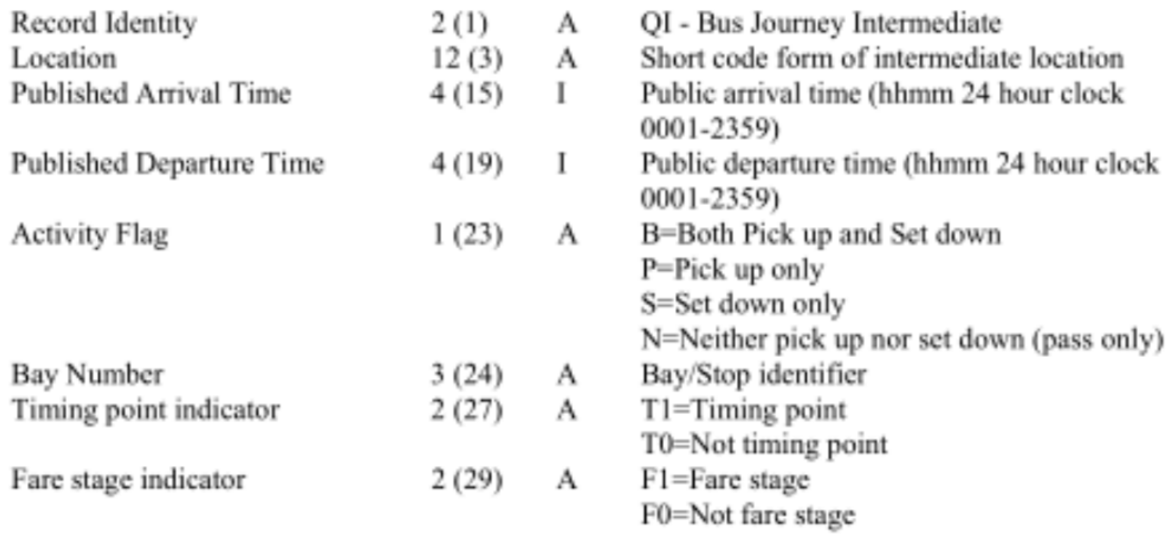
Fig 12 - Journey Origin Record Specification [5]



```
Type        Basemap                                        20210120202101
QSNSTWS1    20201005202010111111100  X74       Bus              I
QO60903723    0915   T1
```

Fig 13 - Journey Origin Record Example [5]

INTERMEDIATE RECORD



Record Identity	2 (1)	A	QI - Bus Journey Intermediate
Location	12 (3)	A	Short code form of intermediate location
Published Arrival Time	4 (15)	I	Public arrival time (hhmm 24 hour clock 0001-2359)
Published Departure Time	4 (19)	I	Public departure time (hhmm 24 hour clock 0001-2359)

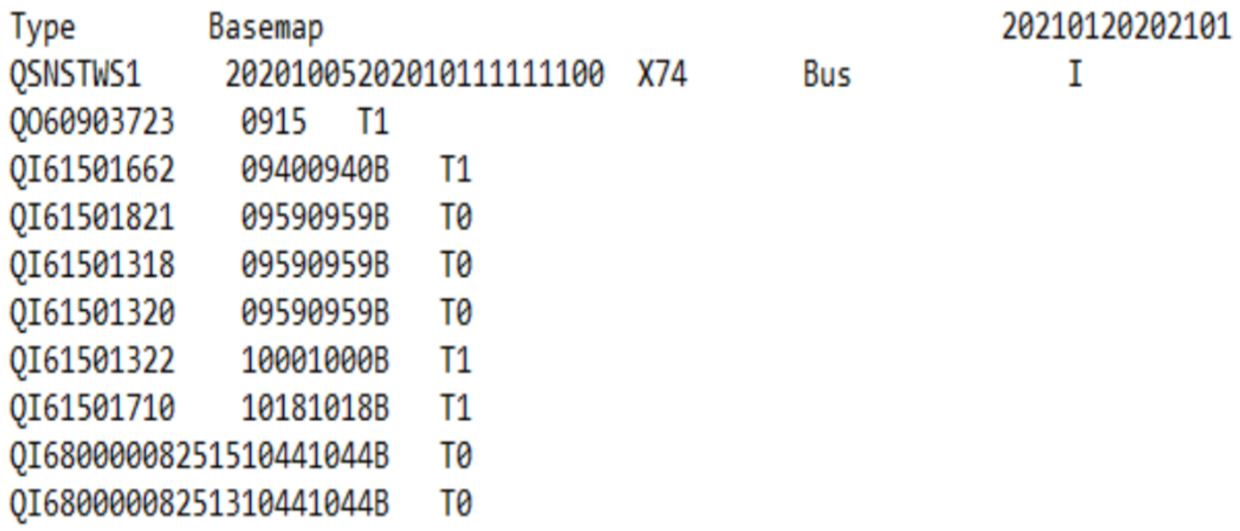
Activity Flag	1 (23)	A	B=Both Pick up and Set down
			P=Pick up only
			S=Set down only
			N=Neither pick up nor set down (pass only)

Bay Number	3 (24)	A	Bay/Stop identifier

Timing point indicator	2 (27)	A	T1=Timing point
			T0=Not timing point

Fare stage indicator	2 (29)	A	F1=Fare stage
			F0=Not fare stage

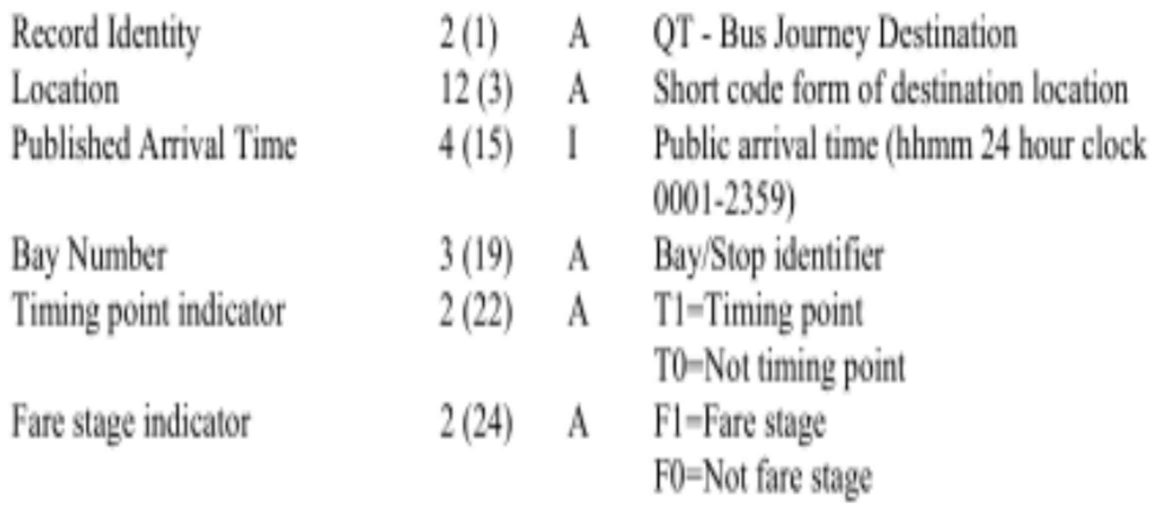
Fig 14 - Journey Intermediate Record Specification [5]



```
Type        Basemap                                        20210120202101
QSNSTWS1     2020100520201011111100  X74       Bus             I
QO60903723    0915   T1
QI61501662    09400940B   T1
QI61501821    09590959B   T0
QI61501318    09590959B   T0
QI61501320    09590959B   T0
QI61501322    10001000B   T1
QI61501710    10181018B   T1
QI68000008251510441044B   T0
QI68000008251310441044B   T0
```

Fig 15 - Journey Intermediate Record Example [5]

JOURNEY DESTINATION RECORD

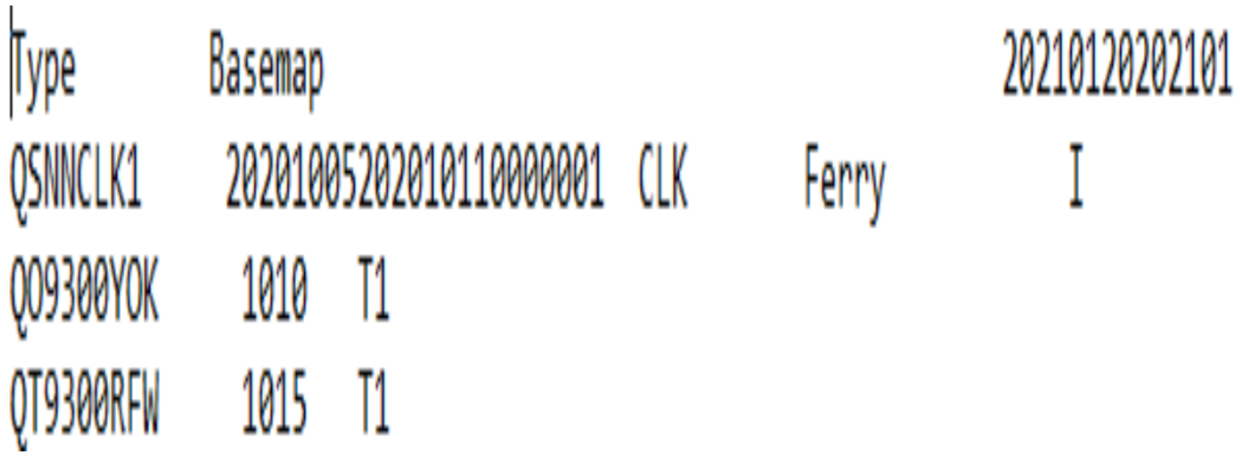


Record Identity	2 (1)	A	QT - Bus Journey Destination
Location	12 (3)	A	Short code form of destination location
Published Arrival Time	4 (15)	I	Public arrival time (hhmm 24 hour clock 0001-2359)
Bay Number	3 (19)	A	Bay/Stop identifier

Timing point indicator	2 (22)	A	T1=Timing point
			T0=Not timing point

Fare stage indicator	2 (24)	A	F1=Fare stage
			F0=Not fare stage

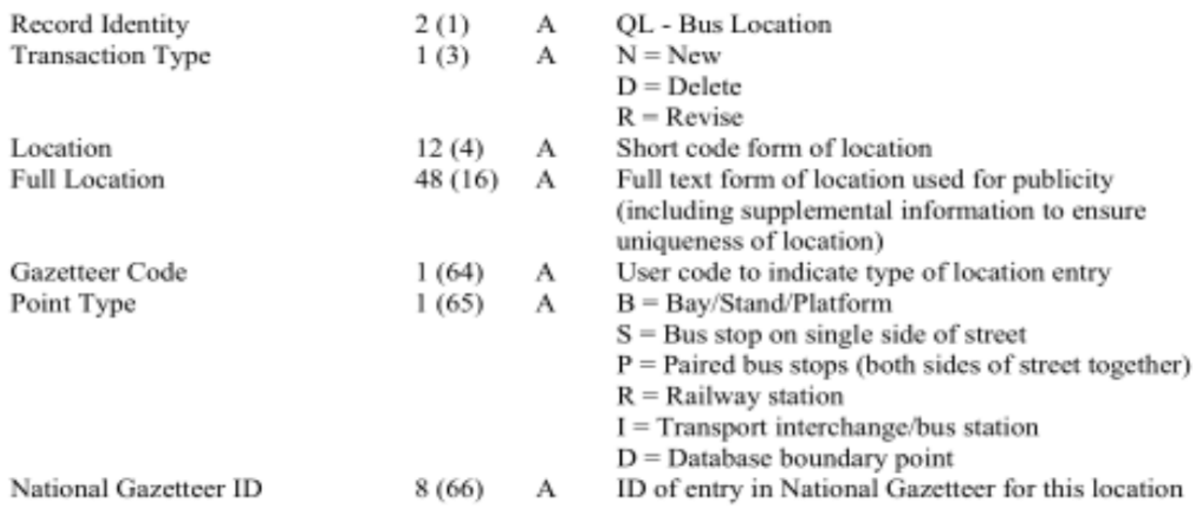
Fig 16 - Journey Destination Record Specification [5]



```
Type       Basemap                                        20210120202101
QSNNCLK1    20201005202010110000001  CLK      Ferry        I
QO9300YOK    1010   T1
QT9300RFW    1015   T1
```

Fig 17 - Journey Destination Record Example [5]

LOCATION RECORD



Record Identity	2 (1)	A	QL - Bus Location

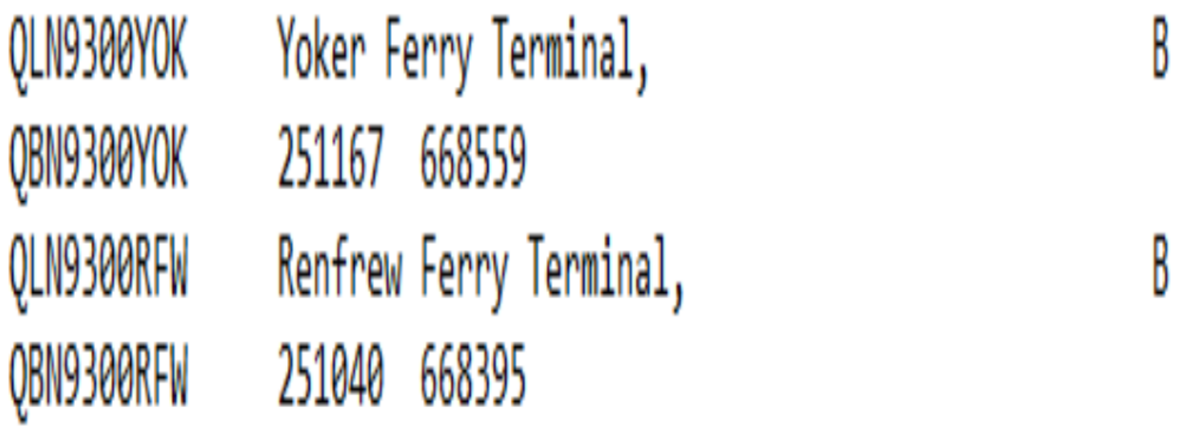
Transaction Type	1 (3)	A	N = New
			D = Delete
			R = Revise

Location	12 (4)	A	Short code form of location
Full Location	48 (16)	A	Full text form of location used for publicity (including supplemental information to ensure uniqueness of location)
Gazetteer Code	1 (64)	A	User code to indicate type of location entry

Point Type	1 (65)	A	B = Bay/Stand/Platform
			S = Bus stop on single side of street
			P = Paired bus stops (both sides of street together)
			R = Railway station
			I = Transport interchange/bus station
			D = Database boundary point

National Gazetteer ID	8 (66)	A	ID of entry in National Gazetteer for this location

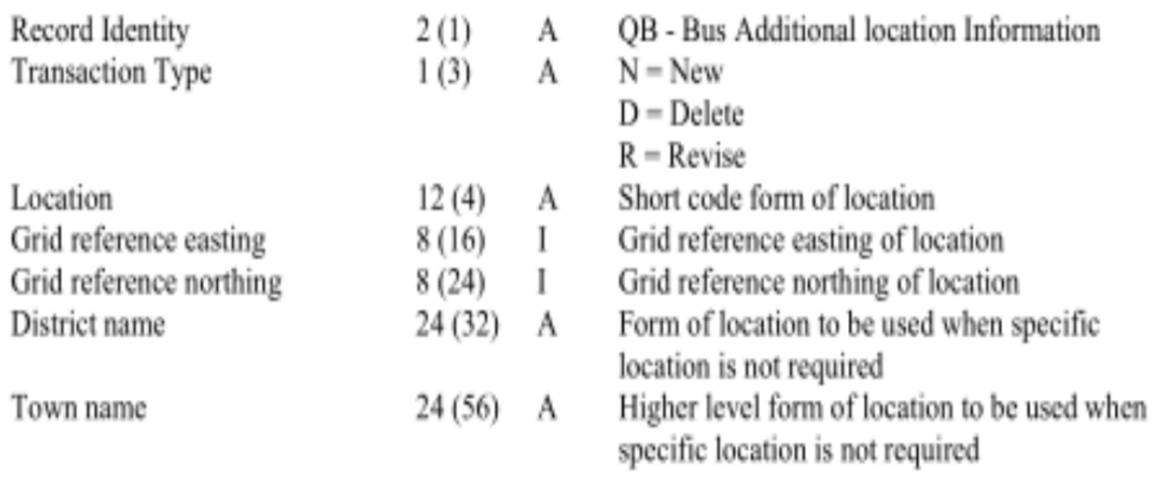
Fig 18 - Location Record Specification [5]



```
QLN9300YOK   Yoker Ferry Terminal,                                    B
QBN9300YOK   251167  668559
QLN9300RFW   Renfrew Ferry Terminal,                                  B
QBN9300RFW   251040  668395
```

Fig 19 - Location Record Example [5]

ADDITIONAL LOCATION RECORD

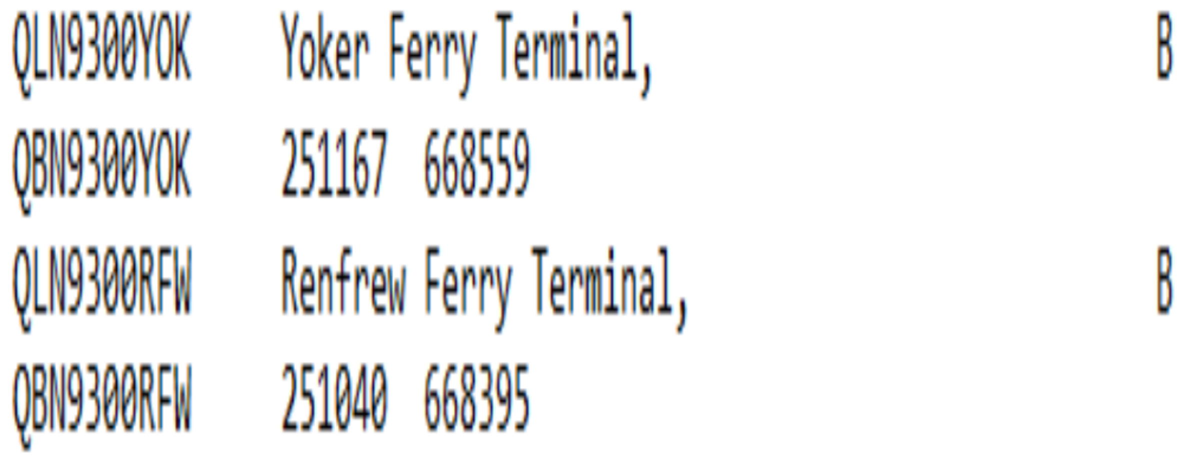


Record Identity	2 (1)	A	QB - Bus Additional location Information

Transaction Type	1 (3)	A	N = New
			D = Delete
			R = Revise

Location	12 (4)	A	Short code form of location
Grid reference easting	8 (16)	I	Grid reference easting of location
Grid reference northing	8 (24)	I	Grid reference northing of location
District name	24 (32)	A	Form of location to be used when specific location is not required
Town name	24 (56)	A	Higher level form of location to be used when specific location is not required

Fig 20 - Additional Location Information Record Specification [5]

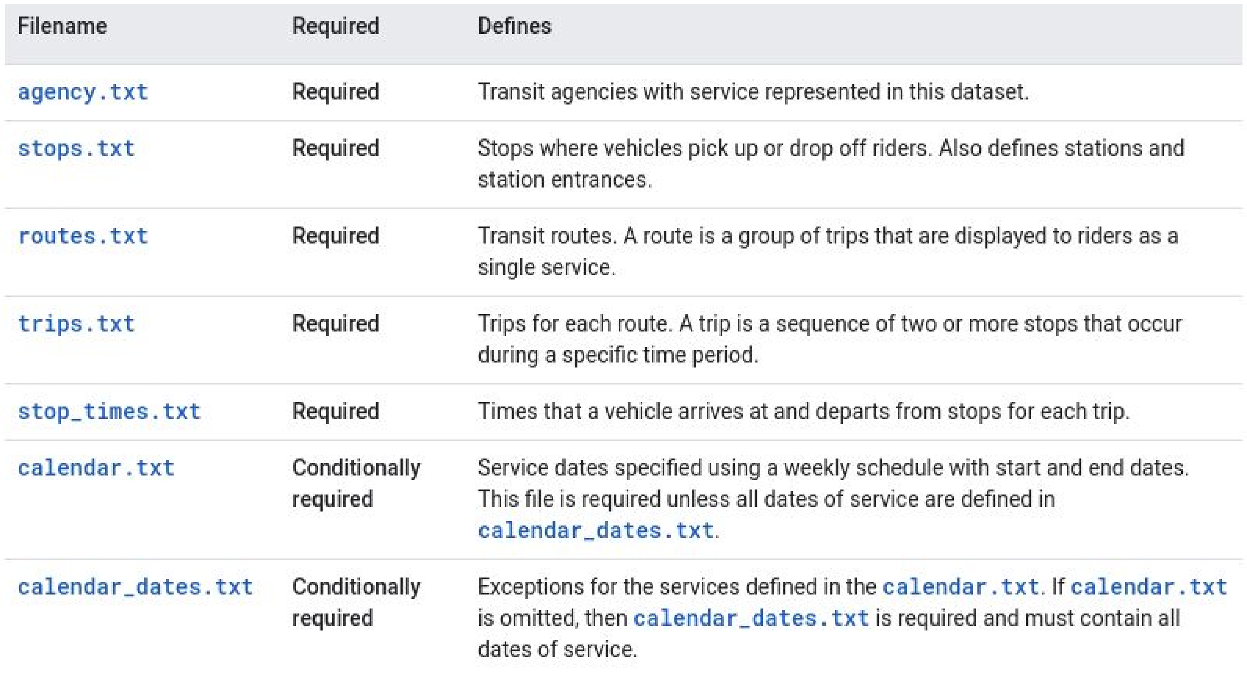


```
QLN9300YOK    Yoker Ferry Terminal,                       B
QBN9300YOK    251167  668559
QLN9300RFW    Renfrew Ferry Terminal,                     B
QBN9300RFW    251040  668395
```

Fig 21 - Additional Location Information Record Example [5]

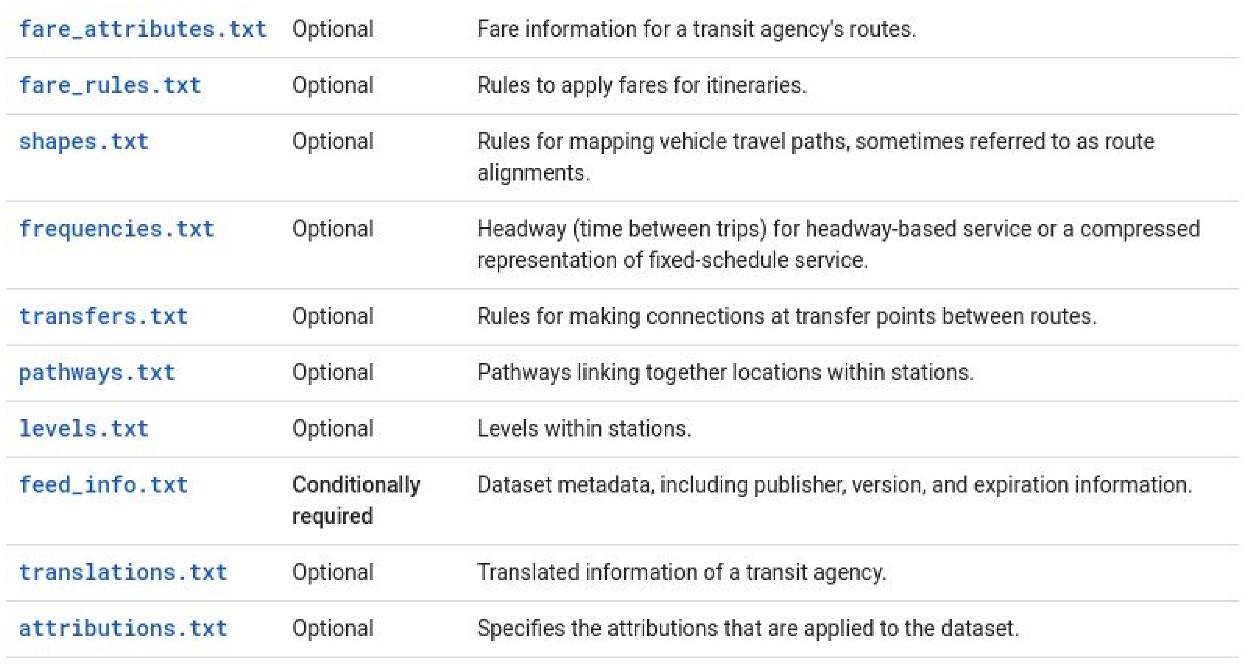
APPENDIX B: GTFS STATIC

FILES IN A GTFS DATASET



Filename	Required	Defines
agency.txt	Required	Transit agencies with service represented in this dataset.
stops.txt	Required	Stops where vehicles pick up or drop off riders. Also defines stations and station entrances.
routes.txt	Required	Transit routes. A route is a group of trips that are displayed to riders as a single service.
trips.txt	Required	Trips for each route. A trip is a sequence of two or more stops that occur during a specific time period.
stop_times.txt	Required	Times that a vehicle arrives at and departs from stops for each trip.
calendar.txt	Conditionally required	Service dates specified using a weekly schedule with start and end dates. This file is required unless all dates of service are defined in calendar_dates.txt.
calendar_dates.txt	Conditionally required	Exceptions for the services defined in the calendar.txt. If calendar.txt is omitted, then calendar_dates.txt is required and must contain all dates of service.

Fig 22 - Table of files in a GTFS Dataset[3]



fare_attributes.txt	Optional	Fare information for a transit agency's routes.
fare_rules.txt	Optional	Rules to apply fares for itineraries.
shapes.txt	Optional	Rules for mapping vehicle travel paths, sometimes referred to as route alignments.
frequencies.txt	Optional	Headway (time between trips) for headway-based service or a compressed representation of fixed-schedule service.
transfers.txt	Optional	Rules for making connections at transfer points between routes.
pathways.txt	Optional	Pathways linking together locations within stations.
levels.txt	Optional	Levels within stations.
feed_info.txt	Conditionally required	Dataset metadata, including publisher, version, and expiration information.
translations.txt	Optional	Translated information of a transit agency.
attributions.txt	Optional	Specifies the attributions that are applied to the dataset.

Fig 23 - Table of files in a GTFS Dataset[3]