Automating Journey Fare Calculation for Transport for London

by

Joshua J. Maciejewski

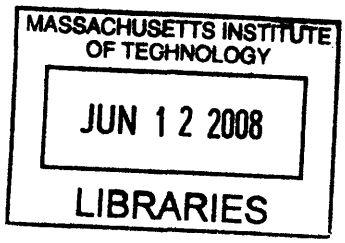
Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Transportation

at the

Massachusetts Institute of Technology

June 2008



Signature of Author .....
Department of Civil and Environmental Engineering
May 9, 2008

Certified by .....
George Kocur
Senior Lecturer of Civil and Environmental Engineering
Thesis Supervisor

Certified by .....
Nigel Wilson
Professor of Civil and Environmental Engineering
Director, Master of Science in Transportation

Accepted by .....
Daniele Veneziano
Chairman, Departmental Committee for Graduate Students

Automating Journey Fare Calculation for Transport for London

by

Joshua J. Maciejewski

Submitted to the Department of Civil and Environmental Engineering
on May 9, 2008 in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Transportation

ABSTRACT

This thesis develops a method to automate journey fare calculation for Transport for London. Today, fares for every possible origin-destination station pair within the London Underground are prepared manually based on the zonal fare structure. Multiple feasible paths often exist within the network for a given origin-destination pair, each of which may produce a different journey fare. Thus, manually adjusting journey fares after any alteration of the network or fare structure is a time consuming task for staff and restricts Transport for London's ability to implement changes in fare policy. This approach also lacks transparency from the passenger's perspective.

Automating Transport for London's fare calculation requires automating the selection of travel paths. This thesis adapts a label-correcting shortest path algorithm to produce journey paths and fares based on four different selection rules: minimum fare, minimum number of transfers, minimum travel time, and minimum distance. The algorithm operates on a directed graph model of the network. This thesis develops a method to structure the directed graph to capture the network's intricacies.

Given a network and fare structure, the modified shortest path algorithm produces all path and fare information for an origin-destination pair in less than one millisecond. Transport for London can then assess the implications of a fare policy change by comparing the existing fares with those generated under each path selection rule. Supplementing these comparisons with historical data provides an estimate of the number of journeys affected and the possible impact on fare revenue. This thesis uses a sample dataset to estimate these impacts.

Thesis Supervisor: George Kocur
Title: Senior Lecturer of Civil and Environmental Engineering

Table of Contents

1 Summary and conclusions 7

1.1 Problem statement 7

1.2 Conclusions 7

1.2.1 Network model and shortest path 8

1.2.2 Tracking variables for path selection and fare calculation 8

1.2.3 Staff and passenger information device 9

1.2.4 Performance 9

1.2.5 Comparison of generated and current fares 9

2 Shortest path algorithms 11

3 Framework 12

3.1 Background 12

3.1.1 Transport for London 12

3.1.2 National Rail 13

3.1.3 TranSys and Oyster 13

3.1.4 Current ticketing and fare structures 13

3.2 Problem statement and research question 17

4 Algorithm 19

4.1 Network model 19

4.1.1 Basic concepts 20

4.1.2 Parallel service case 20

4.1.3 Representing transfers 21

4.1.4 Representing station closures 24

4.1.5 Representing handicap accessibility restrictions 25

4.1.6 Representing two-way service 26

4.1.7 Representing branches 27

4.1.8 Representing out-of-station interchanges 28

4.1.9 Representing intermediate validators 29

4.1.10 Automated arc and node generation 30

4.2 Path selection 33

4.2.1 Minimum fare path 34

4.2.2 Minimum transfer path 34

4.2.3 Minimum duration (travel time) path 35

4.2.4 Minimum distance path 35

4.2.5 Tie breaking 36
4.2.6 Intermediate validation 40
4.3 Tracking variables 41
4.3.1 Tracking visited zones 42
4.3.2 Tracking fares 45
4.3.3 Tracking transfers and last service 50
4.3.4 Tracking duration 51
4.3.5 Tracking distance 51
4.3.6 Tracking predecessor nodes 52
4.3.7 Tracking composite cost 52
4.4 Retrieving path information 53
4.5 Staff and passenger information device 55
5 Database 58
5.1 FareStructure table 59
5.2 FareTypes table 59
5.3 Services table 59
5.4 ServiceLinks table 59
5.5 Stations table 60
5.6 StationTypes table 60
5.7 TransferDurations table 60
5.8 Zones table 60
6 Performance 61
6.1 Performance required 61
6.2 Algorithm performance 62
7 Comparison of generated and current fares 63
7.1 Example data source 63
7.2 Inclusion of London Overground 63
7.3 Minimum fare path 64
7.4 Minimum transfer path 67
7.5 Minimum duration (travel time) path 68
Appendix A: Current TfL fare structure 70
Bibliography 71

1 Summary and conclusions

1.1 Problem statement

This research focuses on automating journey fare calculation for Transport for London (TfL). Today, fares for every possible origin-destination (OD) station pair within the London Underground network are calculated manually based on the zonal fare structure. The manual approach restricts TfL's ability to implement changes in fare policy. Multiple feasible paths often exist within the network for a given OD pair, each of which may produce a different journey fare. Thus, manually adjusting journey fares after any alteration of the network or fare structure is a time consuming task for staff. This approach also lacks transparency from the passenger's perspective.

London Underground fares are a function of the innermost and outermost zones a passenger travels through on a given journey. For example, a journey that begins at a station in zone 3, travels through zone 2, and ends at a station in zone 1, is subject to a zone-1-to-zone-3 fare. Moreover, the same zone-1-to-zone-3 fare applies to a journey that starts in zone 3, crosses central London through zones 1 and 2, and ends at another station in zone 3.

Automating fare calculation requires automating path selection. However, TfL only has knowledge of a passenger's entry and exit locations, and therefore must make assumptions about the actual path of travel. Modeling the TfL network as a directed graph and adapting a shortest path algorithm accomplishes this task.

1.2 Conclusions

The primary conclusion of this research is that automating fare calculation for TfL's network is feasible. One can model the network, with all its intricacies and complexities, in a way that produces the desired results but remains manageable for TfL staff. Adapting a shortest path algorithm automates the selection of journey paths between any two stations. The fare structure and a set of rules guide the algorithm. This algorithm produces the journey fare and a variety of path information relevant to staff and passengers. Thus, the algorithm could function solely as a fare calculator or it might be expanded into an information device for staff and passengers.

1.2.1 Network model and shortest path

Shortest path algorithms operate on the nodes and arcs of a directed graph. Each arc is assigned a cost, which can be defined in a variety of ways. The algorithm finds the lowest cost path from an origin node to any other node in the network. A shortest path algorithm was adapted to produce:

- minimum fare paths,
- minimum transfer paths,
- minimum duration (travel time) paths, and
- minimum distance paths.

Modeling each station as a node and each station-to-station branch of service as a unique arc produces a directed graph representation of TfL's network. The addition of station-service sub nodes and one-way arcs represents transfers between services, or between branches of services. Additional arcs between station entry and exit nodes represent out-of-station interchanges (defined later) and transfers between modes. Entry, exit, or handicap access at represented stations can be disabled for path selection and fare calculation without removing the station for pass-through service.

Assigning each arc's cost under the minimum transfer path requires knowledge of the branch of service associated with each arc and the branch of service utilized to reach the previous node along the shortest path. In the case of the minimum fare path, the cost of each arc must be dynamically determined based on the innermost and outermost zones visited previously in the journey, consistent with the fare rules in London. Transit systems with different fare rules require a different adaptation, though based on the same principles.

1.2.2 Tracking variables for path selection and fare calculation

Whenever the algorithm considers an arc and node, it adds the incremental fare associated with that arc to the journey fare stored for the previous node in the path. Fares are calculated incrementally in this manner, but the fare increment itself depends on two other tracked values: the innermost zone visited and the outermost zone visited. The zone-visited variables are associated with the stations in the journey, not the arcs.

The algorithm tracks transfers using two variables: total transfers made and the previous service used to reach the active node. When the algorithm considers a node and arc, it compares the service associated with that arc to the service used to reach the previous node. If the services are different, a transfer must occur and the arc's transfer increment is one unit.

Tracking variables for distance and duration is straightforward. Each arc is assigned fixed increments of duration and distance. The algorithm tracks total distance and total duration from the root for each node. When a node and arc are considered, the algorithm adds the arc's incremental values to the total values for the previous node.

The algorithm tracks each variable under each path selection method regardless of whether or not it needs the variable to determine the shortest path. Doing so allows the algorithm to produce all relevant path information for comparison of paths selected by the different methods. To eliminate ties between two or more shortest paths, the algorithm makes all decisions based on a composite variable. This composite variable combines the path variables in the order of importance dictated by the path selection method.

1.2.3 Staff and passenger information device

The path selection algorithm tracks many pieces of information about the shortest path that may be useful to passengers and TfL staff alike. A graphical user interface (GUI) could be used to illustrate the shortest path and to provide relevant travel details. A GUI was written as part of this research to serve as an example.

The GUI might be a web-based application accessible to both staff and passengers or restricted to internal use by TfL staff. It might be deployed as an in-station passenger information kiosk or as an application available for download to a PDA. Depending on its implementation, TfL may restrict the options the GUI makes available to passengers.

1.2.4 Performance

Performance of the algorithm is a key requirement for implementation as a real-time component in the fare collection system. The Hao-Kocur algorithm was chosen as the shortest path algorithm for this research because it is arguably the fastest label-correcting algorithm available. Tests conducted on a desktop platform indicate the adapted algorithm is fast enough for use in real-time. The algorithm completes the calculation of the shortest path and all relevant information in less than one millisecond on average.

1.2.5 Comparison of generated and current fares

Automating fare calculation requires that TfL define business rules for determining a passenger's travel path. Depending on the selected combination of rules, generated fares for some OD pairs will

differ from the currently advertised fares. Changes to the graph representation of the physical network, such as deciding to include the London Overground in the LUL fare structure, will also produce different fares.

TfL staff can compare the current fares to fares generated under various decision rules for a given network structure. This allows TfL to identify the OD pairs that would experience a change in fare and, using historical data, estimate the number of journeys affected and the potential impact on revenue. Allowing costless transfer between LUL and the London Overground increases the total number of affected OD pairs and journeys. This effect is more pronounced under a minimum fare path policy than under a minimum transfer path policy because the minimum fare path method is more likely to include the Overground in path selection.

The historical journey sample used for this analysis lacked sufficient detail to produce bankable estimates of each method's impacts on revenue. However, the data indicate that the minimum fare path, minimum transfer path, and minimum duration path methods each result in a decrease in revenue. The average daily pay-as-you-go revenue during the sample period was £1.3 million. The minimum transfer path has the smallest decrease at £32,000 per day, or 2% of pay-as-you-go revenue. The minimum fare path with the Overground included has the largest decrease at £96,000 per day, or 7% of pay-as-you-go revenue.

2 Shortest path algorithms

All shortest path algorithms are labeling algorithms. These algorithms find the path and cost (label) to each node in a network from the origin, or root, node. Shortest path algorithms operate on the arcs out of each node and track candidate nodes on a candidate list. Two general types of shortest path algorithms exist: label setting and label correcting.

Label-setting algorithms, such as Dijkstra's algorithm from 1959, permanently add arcs to the shortest path tree and visit each node only once. Label-correcting algorithms can revisit nodes, which is useful for this research because, in TfL's zonal fare structure, the cost of an arc may change based on the nodes previously visited. Thus, a label-correcting algorithm is a more appropriate choice for calculating minimum fare paths.

Several label-correcting algorithms are available from which to choose. Performance will be critical if the fare calculation algorithm is implemented as a real-time component in a fare collection system. A 1992 report titled *A Faster Implementation of a Shortest Path Algorithm* identified Hao-Kocur as generally the most efficient of the label-correcting algorithms (Hao & Kocur, 1992).

The method of managing and selecting nodes from the candidate list differentiates one label-correcting algorithm from another. Hao-Kocur places a node on the end of the candidate list "only if it has never been on the [candidate list] before and its label is greater than that of the current front node" (Hao & Kocur, 1992, p. 2). In any other case, the algorithm adds the node to the front of the list. More specifically, the Hao-Kocur algorithm operates as follows:

1. Set the root node's label equal to zero and add it to the candidate list. Set the initial label for all other nodes equal to infinity.
2. Select the node on the front of the candidate list and scan each arc out of that node. For each of these arcs, sum the arc's cost with the candidate node's label. Compare this result with the label of the node connected by the arc. If the result is an improvement over the existing label and this node is not on the candidate list, add it to the candidate list in the following manner:
    - If the node was previously on the list, add it to the front.
    - If the node has never been on the list and its label is smaller than the front node's label, add it to the front.
    - Otherwise, add the node to the back of the candidate list.
3. Repeat (2) until the candidate list is empty.

3 Framework

3.1 Background

London's complex transportation system relies on the coordination of numerous entities, both public and private. This section briefly explores the history and role of the entities most applicable to this research. The current fare structure and fare collection system are also discussed.

3.1.1 Transport for London

Transport for London, or TfL, as it is widely known, is a public agency responsible for most transportation-related matters in the city of London, England. TfL was created in 2000 as one of four functional bodies of the Greater London Authority (GLA), which was established by the Greater London Authority Act of 1999. The Metropolitan Police Authority, the London Fire and Emergency Planning Authority, and the London Development Agency are the other three bodies of the GLA (Greater London Authority Act, 1999).

TfL assumed the responsibilities of its predecessor agency, London Regional Transport (LRT), during the 2000 transition. The London Regional Transport Act of 1984 established LRT, and LRT fell under the direct control of the national government through the Secretary of State for Transport (London Regional Transport Act, 1984). London Underground Limited (LUL) was created in 1985, but TfL did not assume direct responsibility for LUL until 2003 due to contract negotiation of a public-private partnership.

The creation of the GLA and TfL marked a significant transition from control by the national government to control by local government. The membership of the GLA includes the Mayor of London and a locally elected 25-member assembly. The mayor also chairs the board of directors that controls TfL. Thus, TfL interprets its task as "to put the Mayor of London's Transport Strategy into action and manage transport services across the Capital" (Company information, 2008). According to the TfL website, implementing this strategy means TfL's responsibilities include:

- London's buses,
- London Underground Limited (LUL),
- Docklands Light Railway (DLR),
- managing Croydon Tramlink,
- 580km of main roads and all of London's traffic lights,

- managing the Congestion Charging scheme,
- regulating the city's taxis and private hire trade, and
- other transportation services (Company information, 2008).

In 2007, through subsidiary companies and private partnerships, more than 1.8 billion passenger journeys were made using TfL's surface network of approximately seven thousand buses, on over 700 routes (London Buses, 2008). More than one billion passenger journeys were made using the LUL network during the same period. The LUL network has over 250 miles of track and nearly 4100 subway cars (London Underground, 2008).

3.1.2 National Rail

National Rail (NR) refers to the private train operating companies (TOCs) created from the break-up of British Rail under the Railways Act of 1993 (About ATOC, 2008). Several TOCs operate passenger suburban rail service in the Greater London area. Numerous London stations serve both the underground and NR, forming a vital link in the commuter network.

3.1.3 TranSys and Oyster

TranSys is a private consortium of Cubic Transportation Systems, Electronic Data Systems (EDS), Fujitsu Services Limited, and WS Atkins (About us, 2008). In 1998, TfL awarded TranSys a contract to design, implement, manage, and market an integrated smartcard fare collection system. Cubic and EDS remain principal partners while Fujitsu and WS Atkins were largely involved in the initial design and implementation of the Prestige contract (Prestige Fact Sheet, 2008).

TranSys developed London's current contactless automatic fare collection system under the name Oyster. Oyster supports preloading of season tickets as well as a prepaid option known as pay-as-you-go. Oyster has been highly successful since coming online in November 2002. Approximately 80% of all bus and Tube journeys are made using Oyster. Over seven million journeys are completed each day using Oyster resulting in ten million daily Oyster transactions (Oyster fact sheet, 2008).

3.1.4 Current ticketing and fare structures

TfL's current fare and ticketing structures vary from mode to mode. Journeys occurring solely on London Bus or the Croydon Tramlink carry a flat fare regardless of the distance travelled. Journeys on the underground, DLR, or London Overground carry a fare based on TfL's zonal fare structure. These

two different fare structures are supported by variations of the ticketing system, which combines magnetic-stripe paper tickets and the Oyster smartcard.

TfL offers a variety of concessionary fares as well as peak and off-peak fares. TfL also introduced a daily capping policy that limits the maximum amount charged to a passenger's pay-as-you-go account each day, regardless of the number of journeys. The following discussions in parts a and b refer to adult peak, single journey fares.

(a) Fares and ticketing on bus and tram

Fares for bus service in the Greater London area were once distance-based, but today all bus and tram journeys carry a flat fare. Valid fare payment for bus and tram includes cash single tickets, Bus Savers (bus only), Bus Passes, Travelcards, and Oyster pay-as-you-go. The fare is due upon boarding, and several routes now require cash single tickets be purchased off-vehicle to expedite boarding (Fares and tickets supplementary information, 2008).

As of January 2008, the cash single fare for bus and tram is £2 and the Oyster single fare is 90p. The significant pay-as-you-go discount reflects TfL's commitment to encouraging Oyster adoption. Travelcards and Bus Passes allow unlimited free travel within a given time period and carry fees commensurate with the length of the period. The current flat fare structure for bus and tram makes fare calculation for these modes straightforward.

(b) Fares and ticketing on LUL and DLR

Valid ticket media for use on the Underground and DLR include cash single tickets, Travelcards, and Oyster pay-as-you-go. Cash singles and short duration (one and three-day) Travelcards are magnetic stripe paper tickets, while pay-as-you-go and longer duration Travelcards require the Oyster smartcard. All passengers must pass through a set of gates at each end of their journey. To gain entry to the Underground, passengers using Oyster tap their smartcard on a card reader attached to the entry gate; exit from the system is accomplished in a similar manner. Many DLR stations are not gated, but passengers are required to tap an Oyster validator at either end of their journeys to mimic passing through the gates. Pay-as-you-go fares are at a substantial discount to cash single fares and are the primary focus of this research.

TfL uses a zonal fare structure to determine pay-as-you-go fares for journeys on the Underground and DLR. This fare structure has nine zones centered on central London. Figure 3.1 shows a portion of the LUL network and the zonal fare structure for northeast London. Zone 1 is the innermost zone. The origin zone and destination zone alone do not determine the fare—fares are calculated based on the innermost and outermost zones *visited* on any given journey. Thus, a trip both originating and terminating in Zone 6 does not necessarily carry a Zone-6-to-Zone-6 fare.

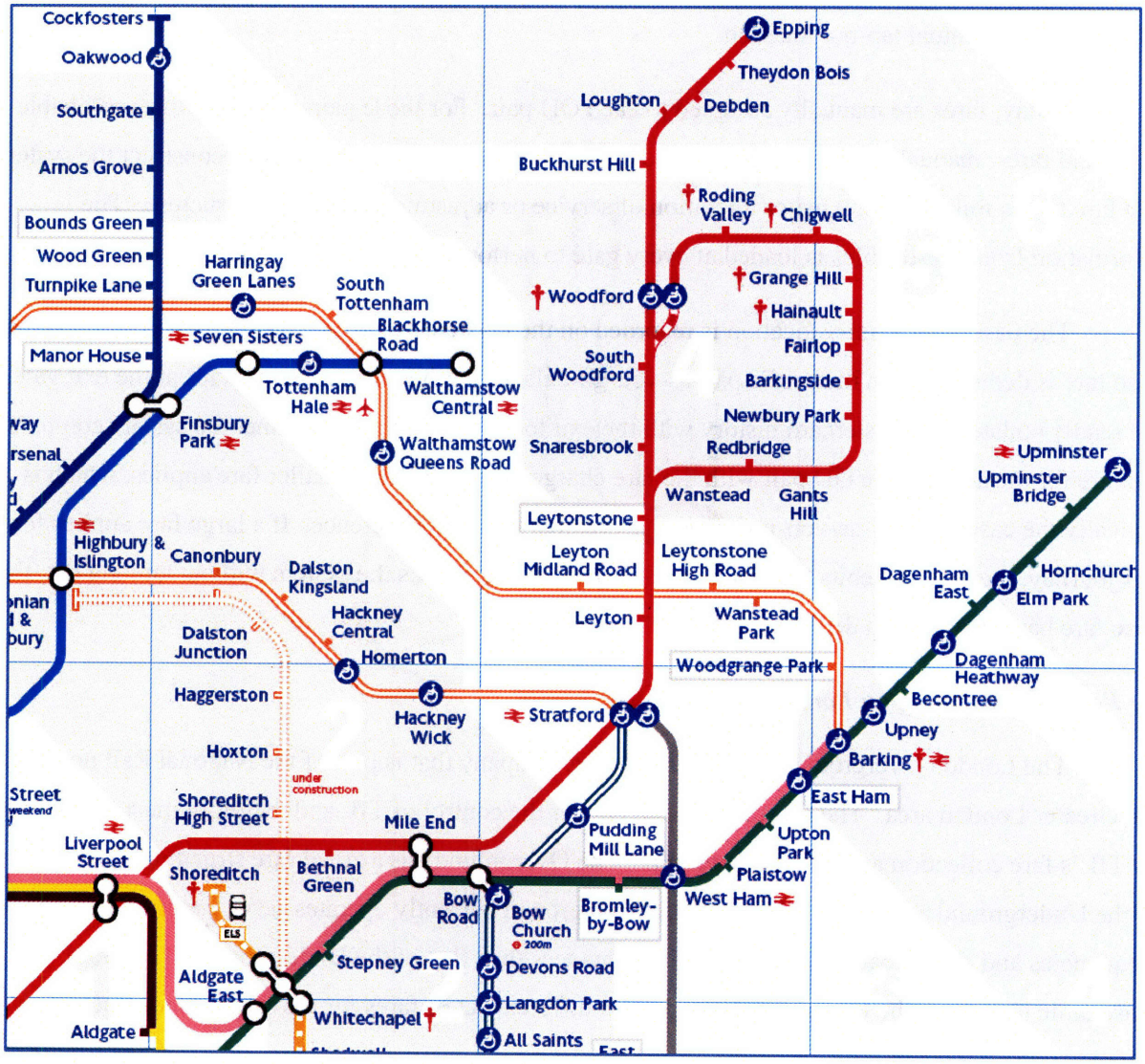


Figure 3.1: This northeast segment of the London Underground map shows parts of TfL's fare zones 1 through 6.

For example, consider a journey from Epping to Upminster. As Figure 3.1 illustrates, both stations are in Zone 6, but no service exists connecting the two stations directly through Zone 6. Thus, the journey fare cannot be a Zone-6-to-Zone-6 fare. A passenger making this journey must take the Central line (red) into an inner zone and transfer to District line (green) service toward Upminster. Numerous

potential paths and transfer locations are available for this journey. The passenger may transfer among several LUL lines, the DLR, or even the London Overground. Depending on the passenger's chosen path, this journey could reasonably require visiting a variety of different zones: Zones 3 through 6 will necessarily be visited, but Zone 2 and Zone 1 might also be visited. Thus, a variety of different fares may be reasonably applicable for the same origin-destination (OD) pair. The fact that most transfers at LUL stations occur behind the gate lines, without record, compounds this issue. Once a passenger taps in and passes through the gate, the system has no further knowledge of the passenger's actual travel behavior other than the eventual tap-out location.

Today, fares are manually assigned to each OD pair. For those pairs with multiple reasonable paths and fares, manual rules are applied to determine the path. TfL and Transys reconstruct the system's OD fare tables following any major alteration of service or adjustment to the fare structure. The information from these tables is loaded at every gate to perform Oyster pay-as-you-go transactions.

The passenger's entry location is recorded on the Oystercard during the entry transaction and a base fare is deducted from the card's pay-as-you-go-balance. When the passenger taps at the exit gate, the reader updates the Oystercard history with the exit location. At the same time, the system compares the fare for the appropriate OD pair with the fare charged on entry. If a smaller fare applies, which is typically the case, the pay-as-you-go account is credited with the difference. If a large fare applies for this journey, the system debits the difference. If the passenger exits the system without tapping out, the base fare has already been deducted.

(c) Fares and ticketing on London Overground

The London Overground is a train operating company that is part of the National Rail network in the Greater London area. The Overground falls under the control of TfL and therefore must be supported by TfL's fare collection and ticketing systems. The Overground has a zonal fare structure similar to that of the Underground and DLR. However, the Overground currently operates service at much lower frequencies and over greater station-to-station distances than that of the Underground. Therefore, it is reasonable to assume the Overground may have a different set of fares for the same zonal structure in the future.

Most transfers between the Underground and the Overground require passengers to pass through an intermediate gate line. The system records these transfers, which can be used for better calculation of fares.

(c) Fares and ticketing on National Rail

National Rail has historically operated on a paper-based ticketing structure with OD-based fares. A growing number of NR services in the Greater London area now accept Oyster-pay-as-you-go fare payment. However, in many cases the NR fare structure is different from TfL's zonal structure and requires special consideration.

(d) Concessionary fares

A variety of discounted fares exists within TfL's current fare structure. For example, 11-15 year old photocards permit free bus travel and reduced LUL and DLR fares for children aged 11 to 15 years. Additional concessionary fares apply to college students, children of various ages, the unemployed, and the elderly. The existence of concessionary fares is largely a political phenomenon and thus TfL's fare collection and ticketing systems must be able to adapt to changing demands.

(e) TfL's evolving approach to fare collection

In an effort to leverage the benefits of technological improvements, TfL created the Future Ticketing Project (FTP). The FTP is concerned with the next generation of ticketing technologies that may be applicable at TfL. Developments in the payment industry may make it desirable for TfL to accept a variety of contactless payment devices at the gate lines including contactless credit cards, key fobs, and NFC-enabled mobile phones. The exploration of alternative arrangements is ongoing. TfL may decide to manage some or all of the existing Oyster fare collection system in-house during a period of transition to a new payment medium.

Many significant changes that will require some adjustment to the fare collection system are occurring within the TfL network; service on the Overground is expanding, more NR services are adopting TfL's payment media, and Crossrail will open in 2017. Additionally, elections for the Mayor of London will take place in May 2008, which may result in notable changes in the general fare structure or concessionary fares. Small adjustments to the base fare structure under the current manual approach would likely require extensive work and may delay implementation of the correct fares.

3.2 Problem statement and research question

This research focuses on a key component of any future TfL-managed ticketing system: automating the calculation of journey fares given a base zonal fare structure and a set of rules. A supporting focus of this work is the development of an automated path selection algorithm for travel within the TfL network. Automation is vital for implementing new fare types, new services, or changes

to the base fare structure. Automation of fare calculation is also essential if TfL chooses to involve a generic acquirer for credit card processing or similar transactions.

Automating fare calculation for TfL's path-based fare policy requires development of a path selection algorithm to choose from among the many feasible routes available between most OD pairs. Modeling the TfL network as a directed graph allows a modified shortest path algorithm to select paths for fare calculation. A set of rules regarding fare path decisions guide this algorithm; TfL will need to establish business rules to define these decisions.

Using well-defined business rules to guide fare calculation will also increase transparency for TfL staff and customers. Today, it is difficult to find an explanation for why a given route is assumed for fare calculation. TfL's online passenger information guide, *Journey Planner*, occasionally suggests travel routes that do not reflect the published fare. Sharing a common path selection algorithm for fare calculation and providing passenger information will eliminate such discrepancies—this research intends to support both fare calculation and passenger information systems.

This thesis first explores how to model the complexities of TfL's network as a directed graph. Then, four different basic path selection rules are explored and implemented within a shortest path algorithm. This algorithm could assist the manual fare calculations performed today; the results produced by the algorithm could be loaded as static fare tables in the Oyster collection system. Under this approach, the speed of the algorithm is not a primary concern. However, computation time becomes a key concern if the algorithm is employed as a real-time component in one or more of the following:

- a future fare collection system,
- a passenger information device, or
- a planning and policy tool.

Therefore, the algorithm's performance is also discussed.

TfL's adoption of business rules to govern path decisions may result in differences for some OD pairs between the fares generated and the fares currently advertised. Changes in the network or fare structure will also produce different fares. Comparing generated and current fares allows TfL to identify the OD pairs affected by a policy change. Historical data can then provide estimates of the number of journeys potentially affected. This thesis uses a sample data set to compare the number of OD pairs and Oyster journeys affected under three of the path choice methods.

4 Algorithm

4.1 Network model

The LUL network, as seen in the standard Tube map in Figure 4.1, can be portrayed as a set of nodes connected by arcs. Algorithms operating on a network of arcs and nodes can identify the shortest path between any two points in the network. This research defines the shortest path in multiple ways including:

- minimum fare path,
- minimum transfer path,
- minimum duration (travel time) path, and
- minimum distance path.

Using an algorithm to produce these paths in the LUL network, with its complex set of services and interchanges, requires a graph representation of the physical network. This section describes how to represent the intricacies of the network for the purpose of fare calculation.

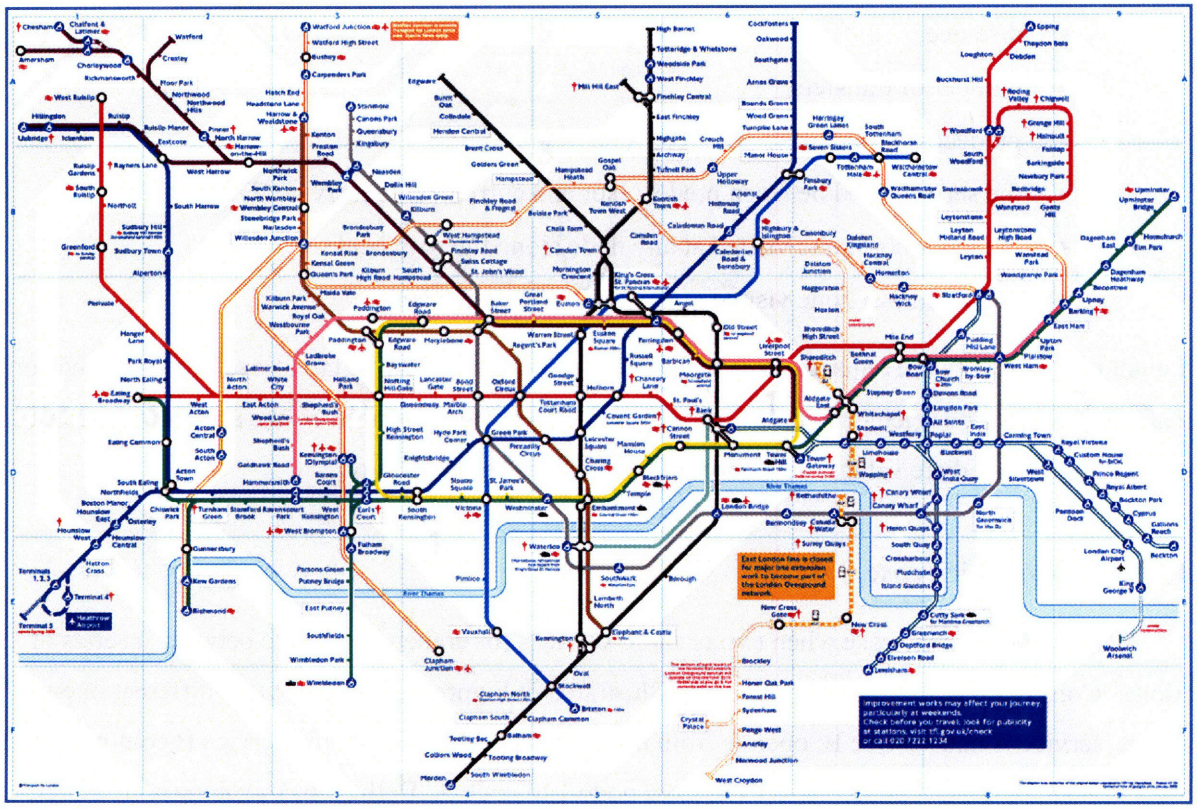


Figure 4.1: This iconic map of the London Underground illustrates the network's complexity (Standard Tube Map, 2008).

4.1.1 Basic concepts

The shortest path algorithm operates on the arcs and nodes of a directed graph. The simplest case is when two or more stations are connected by one service operating in one direction. Modeling this case as directed graph requires treating each station as a node and each connection from one station to the next as a one-way arc, as seen in Figure 4.2.

Figure 4.2: Modeling three stations as nodes and the service connecting them as one-way arcs produces a directed graph.

Assume the passenger starts at a node called the origin, or root, node. For each node in the network, the shortest path algorithm developed in this thesis tracks eight pieces of information to describe the path from the origin node:

- monetary cost (fare),
- duration (travel time),
- distance,
- number of transfers,
- predecessor node (the previous node in the path),
- the service used between predecessor and current node (service ID),
- the innermost zone visited at any node along the path, and
- the outermost zone visited at any node along the path.

All eight pieces of data are required to calculate the paths with the lowest fare, travel time, distance or transfers. The first four elements in this list are direct measures minimized by the algorithm; the last four elements are intermediate values the algorithm uses to support its decisions.

4.1.2 Parallel service case

Next, consider the case when two or more services (or branches) operate between a series of stations. Consider the network of three nodes illustrated in Figure 4.3. Assume two different one-way services, service A and service B, operate from node 1 to node 2 and that only service B continues from node 2 to node 3. Two feasible paths exist from node 1 to node 3. Both paths travel via node 2 and therefore node 3's predecessor will always be node 2 (and node 2's predecessor will always be node 1).

If the fare structure is the same for both of these services, then the fare from node 1 to node 3 will be the same regardless of the path chosen. The innermost and outermost zones visited are the same regardless of path, because either path visits exactly the same set of nodes. The service ID stored for node 3 will be service B, regardless of path choice, because only service B connects node 3 to the rest of the network. However, travel time and distance to node 3 may differ depending on path choice, and the number of transfers required will necessarily differ.

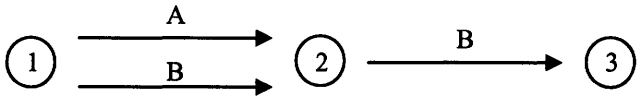


Figure 4.3: Two feasible paths exist for travel from node 1 to node 3 in this three-node network with services A and B.

This journey can be completed by utilizing only service B, resulting in zero transfers, or it may be completed by traveling from node 1 to node 2 on service A and then transferring at node 2 to service B, resulting in one transfer.

Travel time will probably differ based on path choice. If the travel time between nodes 1 and 2 is greater on service B than on service A, and some positive transfer time exists at node 2, there are two potential shortest paths. The passenger can use service B from node 1 to 3, or he can use service A from node 1 to node 2, and then transfer to service B to node 3. If the transfer time at node 2 is less than the extra travel time of service B, the minimum duration path uses both services. Otherwise, the minimum duration path uses strictly service B.

The graph shown in Figure 4.3 cannot model this choice. Therefore, the network representation must be modified to better model transfers.

4.1.3 Representing transfers

The network in Figure 4.4 shows an initial model for transfers:

- split node 1 into two separate nodes, node 1a and node 1b,
- split node 2 into nodes 2a and 2b, and
- add an arc representing transfer from service A to service B via "transfer service" X.

This ensures that the minimum duration path is identified correctly.

The time required for transfer at physical node 2 is the travel time on "service" X from node 2a to node 2b. As long as nodes 1a and 1b are assigned zonal attributes identical to that of node 1, and nodes 2a and 2b are assigned the attributes of node 2, then the innermost and outermost zones visited under any path choice method will be appropriately tracked and fare calculation will be unaffected.

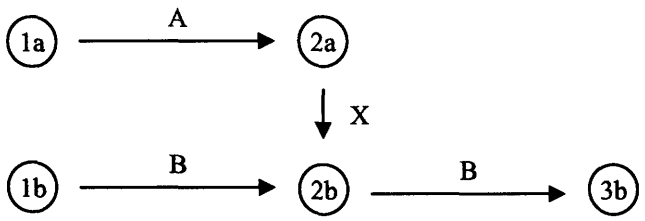


Figure 4.4: Creating a node for each station-service combination allows representation of transfers via service X.

To correctly track transfers between services A and B, a set of rules is implemented to evaluate whether a transfer has occurred with the addition of each node. If the last service used in the path is service X, then a transfer has occurred. Otherwise, the last service must be the same as the current service. However, this approach of splitting each node into service-sub nodes is not sufficient.

The initial origin was node 1 and the initial destination was node 3, not node 1a or node 1b and node 3b. Using the modified graph requires associating the beginning and end of each trip with sub-nodes (like 1a or 1b), rather than the original node (1, or 2). This is awkward.

To avoid this, the graph must include unique nodes representing each station and each station-service combination. Figure 4.5 modifies the three-station example from above to reflect this requirement. An in-station service, service F ("from"), is included to represent travel from the station nodes (nodes 1, 2, and 3) to the appropriate station-service sub nodes (nodes 1a, 1b, 2a, 2b, and 3b). A directed arc of service F is required from a given station node to a station-service sub node whenever travel is feasible from that station on that service.

Service A and service B both operate from station 1, thus two unique arcs of service F must exist: one from node 1 to node 1a and one from node 1 to node 1b. This provides access to each service from node 1. Station 2, however, has only one service operating from it, service B. Thus, only one directed arc of service F is required: from node 2 to node 2b. Station 3 has no services operating from it, and therefore requires no service F arcs.

Similarly, a directed arc of service T ("to") is required from a station-service sub node to a station node whenever that service operates to that station. Nodes 2a and 2b require such arcs to node 2 because both services A and B operate to node 2. Node 3b also requires an arc of service T to node 3 because service B operates to station 3. Node 1 does not require any arcs of service T because neither service A nor B operates to station 1. Defining service F and service T separately is not entirely necessary in this example; service X could be used in place of services F and T as a universal "transfer" service. However, splitting service X into two more descriptive services makes visualization clearer and simplifies the tracking of transfers.

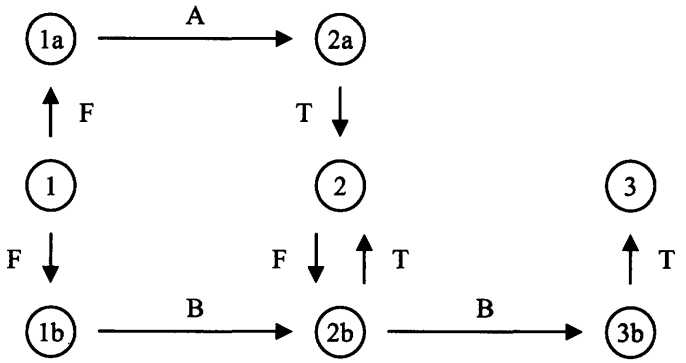


Figure 4.5: Creating a unique node for each station and each station-service combination improves the network model.

If the transfer time between any two services is equivalent for all combinations of services operating at a particular node, then this representation may be sufficient. However, when three or more services operate at a given station this approach is not guaranteed to provide an accurate representation of transfer times.

Consider station 4 with services A and B operating to the station, and service C operating from the station. Assuming transfers are possible from A to C and from B to C, the station and station-service sub node representation would resemble Figure 4.6. If the time required to make each of these transfers is identical, then this model accurately represents the physical network. Transfers must occur through node 4, but since the transfer times are identical, this time can be assigned to the arc with service F. A logical operator within the shortest path algorithm then ensures the transfer time is counted only if the passenger is transferring and that it is not counted if the journey is originating at node 4. However, if the transfer times are different in each case, the logical operations required to assign transfer duration to the arc with service F become complex and unmanageable.

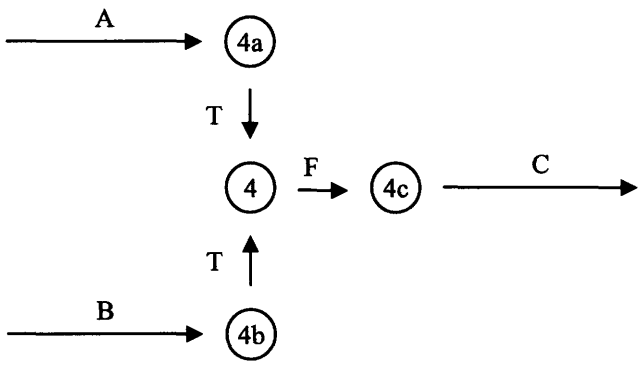


Figure 4.6: Accurately representing transfer times at stations with more than two services is difficult under this network model.

A more robust approach creates a unique arc directly between each pair of station-service sub nodes for every possible transfer within a station. Figures 4.7 and 4.8 reflect this approach, which allows straightforward assignment of the appropriate transfer times for every feasible combination.

4.1.4 Representing station closures

Splitting node 4 into source node 4I and a sink node 4E, representing ingress to and egress from the system, respectively, helps addresses the transfer issue and improves functionality. Physical stations may close for passenger entry and/or exit while remaining operational for pass-through services. For example, some stations such as Turnham Green via the Piccadilly line, allow entry and exit only during certain periods, but allow pass-through service continuously. Treating passenger entry and exit as two separate nodes for each station provides the ability to represent closures.

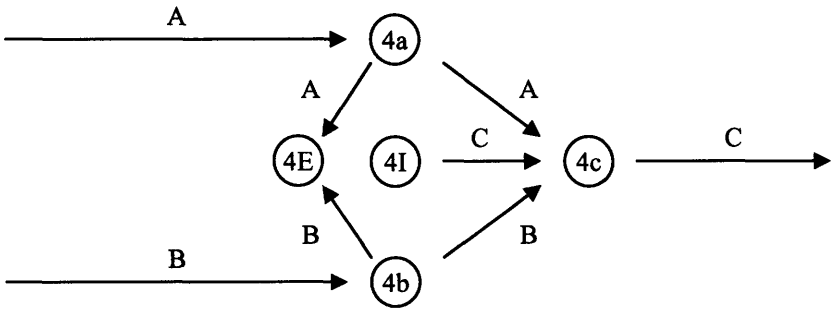


Figure 4.7: The addition of transfer arcs between station-service sub nodes improves the representation of transfers.

Thus, even though the graphical representation grows more complex through the addition of more nodes and arcs, the overall calculation of the shortest path becomes more intuitive and efficient. Figure 4.7 revisits the case of station 4 under this new representation, and Figure 4.8 similarly revisits the three-station case described earlier.

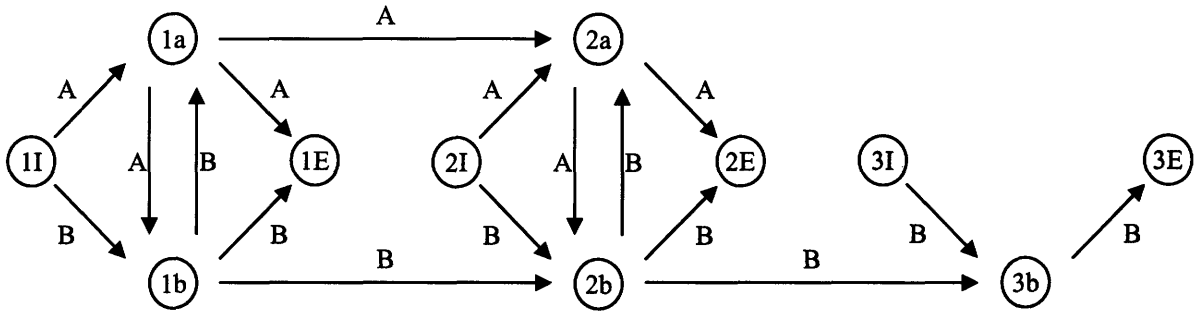


Figure 4.8: Adding entry and egress nodes at each station ensures transfers will use the direct transfer arcs and also helps support station closures.

4.1.5 Representing handicap accessibility restrictions

Most of LUL's stations were constructed at a time when accessibility by persons with disabilities was not a primary design consideration. TfL has undertaken renovation projects to improve accessibility, but many stations remain inaccessible. Some stations with large gaps between platform and train are completely inaccessible for persons with disabilities. Others are accessible for transfers from one service to another, but cannot be used for entry or exit from the system because of stairwells.

Representing these accessibility restrictions is desirable when employing the path selection algorithm as part of a passenger information system. Handicap accessible routes for an OD pair can be produced upon demand. Altering the graphical representation allows incorporation of the accessibility constraints:

- If a station is not handicap accessible for entry and exit, do not create the ingress and egress nodes at that station. This is similar to disabling a station for entry and exit, but it will only occur when requesting handicap-accessible routes.
- If a station is not handicap accessible for transfers, do not create transfer arcs between any of the station-service sub nodes.

Removing the entry and exit nodes for a given station necessarily disables this station as an entry or exit location. Removing the transfer arcs necessarily prevents transfers or off-vehicle intermediate validations at this station.

4.1.6 Representing two-way service

The examples earlier in this chapter assume travel on each service occurs in one direction: from left to right. In a directed graph, representation of two-way service between any two adjacent stations requires two unique one-way arcs.

Under the station-service sub node representation, these two arcs can both be constructed between the same two station-service sub nodes, as seen in Figure 4.9. The shortest path algorithm will track each data item correctly using this model, and each of the path choice methods will produce the correct paths for the constructed graph, with one exception.

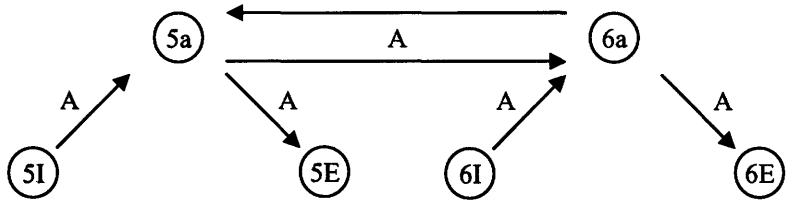


Figure 4.9: Representing two-way service requires two direct arcs for each station-to-station connection. Figure 4.10 shows a more robust representation.

A problem arises when a transfer occurs from service A-leftward to service A-rightward, or vice versa. From the algorithm's perspective, a transfer has not occurred because the two arcs are both of service A and no intermediate service was used. The algorithm will not increase the total travel time or the transfer count. Figure 4.10 shows a representation that handles this issue correctly.

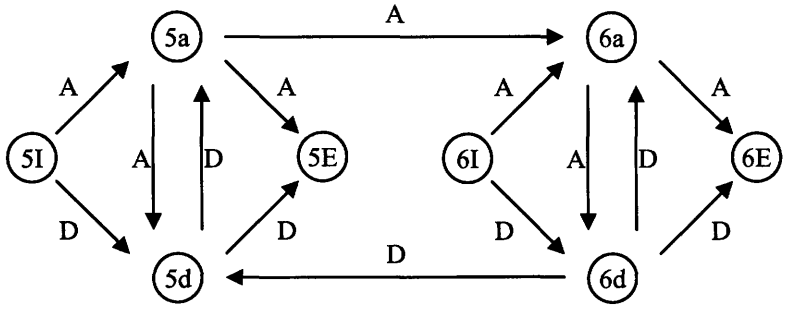


Figure 4.10: Representing two-way service as two separate services (A and D) improves path description and transfer time tracking.

Although it does not affect the path choice or fare calculation, another issue develops from the representation in Figure 4.9 when producing a text description of a generated path. Using this model makes it difficult to provide a meaningful path description for passengers and planners because the name of a given service, which will be reported in the path description, will be the same regardless of the direction of travel. Therefore, it is desirable to construct additional station-service sub nodes (5d and 6d) and define a new service (service D) to separate the representation of a physical two-way service, as seen in Figure 4.10. Service A becomes the eastbound service and can be given a more descriptive name, such as "service A eastbound". Service D, which represents the westbound direction of service A, can be given the name "service A westbound."

4.1.7 Representing branches

Many LUL services, such as the Piccadilly line, have more than one branch. Some stations are served by only one branch; others are served by multiple branches. For example, only the Heathrow branch of the Piccadilly line serves South Ealing, but both the Heathrow and Uxbridge branches serve Acton Town. In the opposite direction, services from either branch terminate at Cockfosters, the end of the Piccadilly line. Figure 4.11 displays this section of the Piccadilly line.

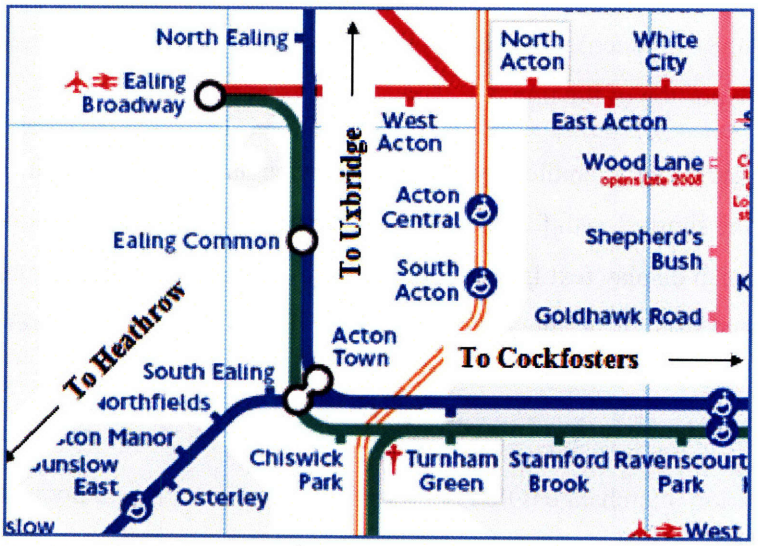


Figure 4.11: The Piccadilly line (blue) branches at Acton Town.

Every branch is represented as a separate service in the model for the same reasons two-way service is represented with two arcs. Additional station-service sub nodes and transfer arcs are created as necessary so that transfers are modeled correctly.

Piccadilly line services operating on either branch toward Cockfosters eventually travel through the same trunk portion of the Piccadilly line. Thus, a journey from Uxbridge to Cockfosters on the Piccadilly line may be completed without transferring branches or services. Similarly, a transfer-free journey is possible from Heathrow to Cockfosters or from anywhere else along the Piccadilly line to Cockfosters. This is represented by modeling the Piccadilly line as three separate services: one toward Uxbridge, one toward Heathrow, and one toward Cockfosters. A series of directed arcs of Piccadilly line service toward Cockfosters start at both Uxbridge and Heathrow, sequentially connecting each station along the respective branch until converging at Acton Town. A single series of directed arcs continues from Acton Town to Cockfosters. Conversely, the two branches of the Piccadilly line operating from Cockfosters cannot share the same directed arcs along the trunk portion of the line. Each branch must be represented separately for every station it serves or the algorithm may identify a transfer where one is not necessary.

Representing the branches in this way solves the issues of path generation for any OD pair along the Piccadilly line, and it solves the problems of path description for journeys commencing or terminating in the non-trunk sections of the Piccadilly line. However, for paths both originating and terminating along the trunk section of the Piccadilly line heading away from Cockfosters, the algorithm produces an undesirable result. The text description of the path will instruct passengers to use the Uxbridge branch (or equivalently, the Heathrow branch) when they could actually use either because their journeys do not deviate from the trunk portion of the line.

This issue is most easily handled by keeping a service list for origin-destination pairs having multiple services. If the shortest path for one of these origin-destination pairs uses one of the services in the list, the algorithm can display text instructions that any of the services may be used. This list can be automatically generated from the graph and does not alter the structure of the representation.

4.1.8 Representing out-of-station interchanges

An out-of-station interchange (OSI) is a transfer from one service to another that differs from an ordinary transfer. As the name implies, an OSI is a transfer that is not conducted solely inside a station. The two stations involved in an OSI, although usually geographically close, are not connected by a common service. To complete an OSI, the passenger must pass through the gate line of one or both stations, and must tap his or her payment device accordingly.

In order to represent the possibility of a transfer between two stations, these stations must be connected with one or more directed arcs in the direction transfers are possible; two directed arcs, one beginning at each station, are necessary between the two stations if an OSI is possible in both directions. A unique service is defined to identify these arcs. The universal OSI service is used whenever an OSI is possible. Since each station is represented by a set of sub nodes rather than a single node, it is important to associate the OSI arcs with the appropriate nodes. If an OSI is possible from station 7 to station 8, then a directed arc of the universal OSI service must be constructed from the egress node of station 7 to the ingress node of station 8, as seen in Figure 4.12. This will enable transfers from every service operating at station 7 to every service operating at station 8. An appropriate value is assigned to this OSI arc to reflect the duration of the transfer. If a transfer is possible in the opposite direction, from station 8 to station 7, then an additional OSI arc is required from the egress node of station 8 to the ingress node of station 7.

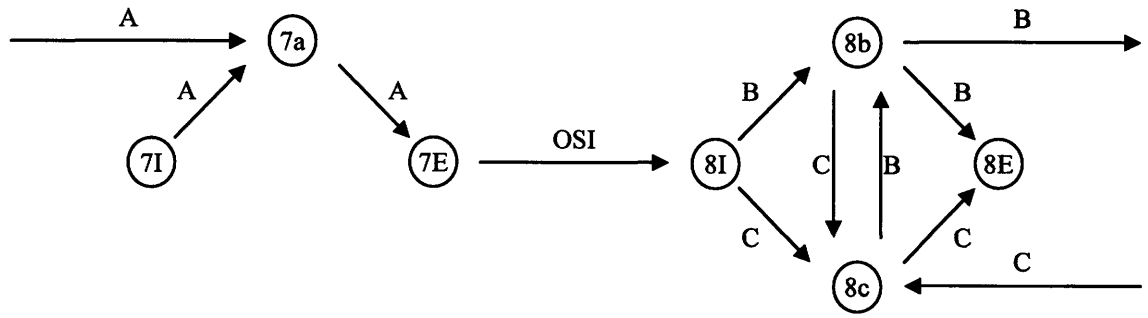


Figure 4.12: A directed arc of service "OSI" represents an out-of-station interchange from station 7 to station 8.

The shortest path algorithm will recognize that a transfer is possible between these two stations and will produce the correct paths under each path choice method. When the algorithm is used as a fares engine to calculate a fare given a sequence of tap-ins and outs, the OSI taps can be treated as intermediate validations in the journey, the intermediate segment being the short OSI leg between stations 7 and 8.

4.1.9 Representing intermediate validators

In the physical network, an intermediate validation may occur whenever a payment device reader is available behind the gate line. Such a device is used to provide information to the fare collection system regarding a passenger's travel behavior. Intermediate validation occurs today in London in a variety of cases including out-of-station interchanges, transitions from the Underground to the DLR, or

transitions from the Underground to tram services. Future fare policy may allow TfL to adjust a given journey's fare based on a passenger's intermediate validation behavior.

At TfL, these payment device readers are known as a passenger validators, or PVs. Similar to a station exit, a PV can be represented in a directed graph as an additional node made accessible from every station-service sub node via one-way arcs. Although a validator node does not represent exit from the system, it must be treated as a terminal node because it cannot be the origin of an arc. If an arc originating at a validator node and terminating at a station-service sub node were added to the graph, transfer durations would no longer be accurately represented; transfers would occur through the costless path provided via the validator node rather than along a transfer arc. A discussion of how intermediate validation behavior may be modeled using this graphical representation is contained in section 4.2.6.

4.1.10 Automated arc and node generation

Representing TfL's transit network requires a graph of sub nodes and adjoining arcs for every station. Manually constructing the representation of a station with multiple services is cumbersome. The standard Tube map published by TfL for passengers includes more than 300 stations, most of which are connected by multiple services. This map does not include the growing number of National Rail stations surrounding London that now support fare payment via TfL's fare collection systems.

Manually creating and editing every sub node and arc in the graphical representation of this system is, at best, burdensome and is a likely source of error. Overlooking the creation of one sub node or a single arc may result in the production of erroneous paths and fares. Isolating the source of the error among hundreds, or thousands, of arcs and nodes promises to be time consuming. The ability to quickly alter the network representation following a change in fare policy or a change in the physical network was a primary motivation for automating fare calculation. Thus, the manager's interaction with the system must be as straightforward and intuitive as possible. An algorithm was developed to eliminate the need to explicitly define each supplementary sub node and arc, which greatly reduces the manger's workload and simplifies data storage. This algorithm utilizes manually defined stations, services, transfer times, and service links to generate a complete graphical representation of the physical network.

Section 5 describes the data required to generate the graph. This includes defining:

- each station and its attributes,
- each direction of each branch, of each physical service, and
- each physical station-to-station service link.

Each station-to-station service link includes the origin station ID, the destination station ID, the service ID, the duration of travel, and the distance of travel. The algorithm creates all of the sub nodes and arcs to represent the physical network based on the stored data. This process occurs in three steps:

- construct ingress and egress nodes,
- construct station-service sub nodes, and
- construct transfer and station-to-station arcs.

(a) Construction of ingress and egress nodes

The automated construction of the network begins with the creation of the ingress and egress nodes as follows:

- For every station with entry enabled, an ingress node is created. For every station with exit enabled, an exit node is created. If handicap accessibility mode is enabled, ingress and egress nodes are created only at stations that are handicap accessible and have entry and/or exit enabled.
- For every station with intermediate validation enabled, a validator node is created.
- Each of these nodes is assigned the zonal and map coordinate attributes of its associated station.
- Each node is assigned a unique ID; the first node created is assigned node ID 0 and each additional node is assigned a sequential ID. The node ID is the most important attribute for identifying the node in the graph and it will typically differ from the associated station ID.

A two-dimensional array tracks the nodes associated with a given station. The first dimension of the array identifies the station and the second dimension identifies the service (with ingress and egress each treated as unique, automatically defined services). The generated node's ID is stored in this lookup array.

(b) Construction of station-service sub nodes and station-to-station arcs

Once the algorithm has created an ingress, egress, and validator node for every enabled station, it begins construction of the arcs and station-service sub nodes. The algorithm performs the following steps for each service link:

1. If an appropriate station-service sub node does not already exist, create one at the origin station and assign this node the zonal attributes of its station.

2. If an appropriate station-service sub node does not already exist, create one at the destination station and assign this node the zonal attributes of its station.

3. Create an arc, with service ID equal to that of the service link, representing entry from the origin station ingress node to the origin station-service node in (1) above. Assign zero duration and distance values to this arc.

4. Create an arc from the origin station-service node in (1) to the destination station-service node referred to in (2). Assign this arc the service ID, duration, and distance attributes defined for the service link.

5. Create an arc, with service ID equal to that of the service link, from the destination station-service node referred to in (2) to the destination station egress node. Assign zero duration and distance values to this arc.

(c) Construction of transfer arcs

Once the algorithm has transformed every service link into sub nodes and arcs, it constructs the appropriate transfer arcs as follows:

- For each station with an intermediate validation node, a single one-way arc is constructed from every station-service sub node to the station's intermediate validation node. Each of these arcs is assigned the service ID of the station-service sub node from which it originates. Each arc is also assigned zero duration and zero distance.
- For every station with two or more station-service sub nodes defined, the algorithm creates two arcs (one in each direction) between each pair of station-service sub nodes.
- Each of these transfer arcs is assigned the service ID of the service from which it is transferring and a distance of zero.
- Each transfer arc is assigned a duration in one of two ways:
    - If the stored data specifically defines a duration for this transfer at this station (in the TransferDurations table), the algorithm assigns this value.
    - If a duration is not defined for this specific transfer, the algorithm assigns the value of the station's default transfer attribute.

Assigning transfer duration in this way allows transfers between different services at the same station to accurately reflect physical reality. Transferring from District to Circle line service at Westminster station, for example, occurs on the same platform and thus carries a duration approximately equal to the headway between trains of either service. However, transferring from the District line to the Jubilee line at

Westminster has a greater duration because the Jubilee line operates approximately 100 feet below the District line.

The graph is now fully constructed. The algorithm does not alter the stored data during this process, and the graph is regenerated based on the stored data each time the algorithm is initialized. Thus, although the assignment of station IDs is critical for interaction with the stored data, the actual values of the station IDs are inconsequential because the application generates node IDs sequentially from zero. As long as the station IDs are consistent throughout the stored data, the arc and node generation algorithm will associate every generated node with the appropriate station and every arc with the appropriate nodes. This allows each stored station ID to take on any unique integer value, which is especially useful when deleting a stored station.

4.2 Path selection

Modeling the LUL network as a directed graph allows the use of a shortest path algorithm to determine journey paths and fares. Shortest path algorithms operate based on the utilization cost assigned to each arc in the graph. It searches for the shortest path from a given origin or root node based on these costs. This research defines cost in different ways depending on the path selection criteria. Four different path selection methods were developed and explored:

- minimum fare path,
- minimum transfer path,
- minimum duration (travel time) path, and
- minimum distance path.

This section describes each of these methods.

For many OD pairs, multiple paths exist as a solution for each path selection method. Ensuring the algorithm appropriately chooses one path from among a set of feasible solutions requires a process for tie breaking, which section 4.2.5 describes.

TfL may wish to offer multiple fares for a given origin and destination based on more than one path selection method. For example, TfL may charge the fare associated with the minimum duration path by default, but if the minimum fare path's duration is reasonably close to the minimum duration, TfL may offer the cheaper fare to a passenger if he proves he traveled via the cheaper route. Providing such proof will likely require the use of intermediate validation for the production of non-standard travel paths, which section 4.2.6 describes.

4.2.1 Minimum fare path

The minimum fare path is the cheapest possible path from an origin station to a destination station. Cost of travel on each arc in the graph takes on a currency value (in TfL's case GBP) representing the incremental increase in journey fare if a given arc is utilized. The minimum fare path method seeks the cheapest possible route between the desired OD pair. It ignores the number of transfers, duration, or distance of a journey except for the purpose of tie breaking, as discussed in section 4.2.5. The algorithm always tracks these additional data items for reporting purposes, but they are not primary considerations under the minimum fare path.

In a zonal fare structure such as TfL's, the incremental fare for traveling on an arc depends on several factors including each connected station's zone and the zones already visited during the journey. Thus, the cost assigned to a given arc may change with each visit of the algorithm—this is why a label-correcting algorithm is the most suitable option for automated fare calculation at TfL. The algorithm determines a given arc's fare increment by comparing the applicable fare for the innermost and outermost zones already visited in the journey with the applicable fare for the innermost and outermost zones visited after the addition of the arc.

Consider an arc connecting the two adjacent stations Liverpool Street in Zone 1 and Bethnal Green in Zone 2. The incremental cost of travel on this arc depends on where the passenger's journey began, or more specifically the zones already visited in this journey. If the passenger has only traveled in Zone 1 thus far, the incremental cost of travel on this arc is the difference between a Zone-1-to-Zone-2 fare and a Zone-1-to-Zone-1 fare. However, if the passenger's outermost zone already visited is Zone 3 and innermost is Zone 2, then the cost of travel on this arc is now the difference between a Zone-1-to-Zone-3 fare and a Zone-2-to-Zone-3 fare.

The incremental cost of an arc connecting stations in two previously visited zones is costless because the fare does not change. Similarly, after the first arc entering a zone, travel on subsequent arcs between stations of the same zone is costless from the algorithm's perspective. Appropriate tracking of visited zones is essential for determining the minimum fare path—section 4.3.1 provides an overview of tracking visited zones.

4.2.2 Minimum transfer path

The minimum transfer path is the path from origin to destination requiring the fewest number of inter-service, and inter-branch, transfers. The minimum transfer path method ignores the duration of the journey and all other cost implications; it chooses a path with the fewest possible transfers between

services or branches of services. This path is often not unique, as multiple paths with the minimum number of transfers may exist.

The algorithm determines the incremental cost of travel on an arc by whether or not the arc's service ID is the same as the last service used to reach the predecessor node. If the services are the same, travel on this arc is costless; if the services differ, this arc has a cost of one transfer unit. The arcs automatically generated to represent transfers between station-service sub nodes, as described in section 4.1.10, are assigned the service associated with the station-service sub node from which they originate. Thus, the algorithm does not assign a transfer cost when utilizing a transfer arc. The algorithm imposes a transfer cost when the subsequent arc is utilized, given the subsequent arc originates at the station-service sub node at which the transfer arc terminates.

4.2.3 Minimum duration (travel time) path

The minimum duration path is the path from origin to destination that has the smallest expected travel time. The total travel time considered includes the expected duration of travel between each station as well as the transfer time between services, when applicable. The incremental cost of travel on any arc is therefore the expected duration of travel on that arc. The duration assigned to an arc connecting station-service sub nodes of two different stations is the expected in-vehicle travel time. Dwell times, station ingress, and station egress times were not explicitly included this research, but slight modifications to the network representation could incorporate all three.

The arcs automatically generated to represent transfers between station-service sub nodes, as described in section 4.1.10, are assigned the expected time for the transfer including walk and any wait time. A transfer made between two services, or between two branches of the same service, must utilize one of these arcs. In doing so, the duration assigned to that arc will be appropriately included in the calculation of journey duration.

4.2.4 Minimum distance path

The minimum distance path is the path from origin to destination with the smallest total in-vehicle distance traveled. The incremental cost of an arc connecting two nodes from different stations is the distance traveled between the two stations. Transfer arcs and station ingress and egress arcs are assigned zero incremental distance costs. The algorithm works equally well defining distance as in-vehicle travel distance or as some other measure of distance (map distance, for example). The minimum distance path may not be unique.

4.2.5 Tie breaking

The structure of each of the basic path choice methods gives rise to the possibility that multiple paths will exist meeting the given minimization criteria. The existence of multiple feasible paths is a matter for concern from both a fare calculation perspective and an information system perspective. Under the minimum fare path method, the algorithm will always produce the correct fare, but the path assumed and recommended may needlessly have a travel time, distance, or number of transfers, greater than a different path producing the same minimum fare.

The selection of one path from among multiple feasible solutions under each of the other path methods is important because the selection may also influence fare calculation. The basic path choice methods will not always choose the most logical path from among a set of tying solutions—they must be altered to include secondary and tertiary selection criteria. Defining and tracking a new, unit-less variable representing composite costs presents an efficient mechanism for tie breaking.

(a) Tie breaking under minimum fare path

The basic structure of the minimum fare path method ignores durations and assigns zero incremental cost to a transfer between services. This allows the algorithm to seek out the cheapest possible path, ignoring non-monetary costs such as passenger inconvenience or travel time. However, as a result, the minimum fare path method may produce paths with needless transfers or longer-than-necessary durations, which is undesirable from a planning perspective for both travelers and policy makers.

A journey from South Kensington to Westminster, as seen in Figure 4.13, produces the same fare if the path includes multiple transfers between two services as it does if the path utilizes only one service.

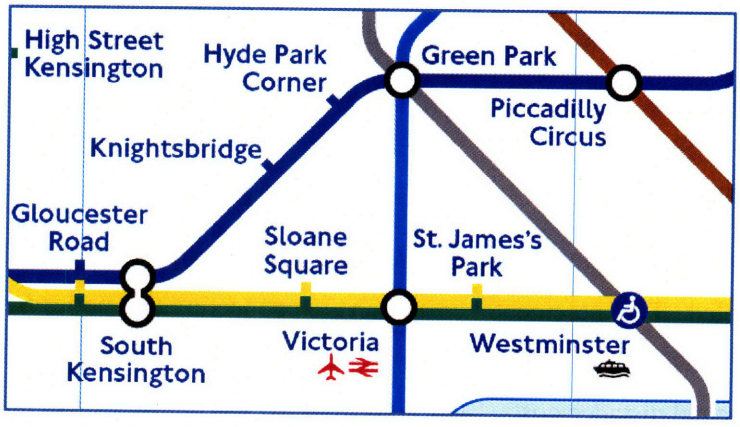


Figure 4.13: The minimum fare path method may produce unnecesarry transfers for travel between stations with parallel services, such as from South Kensington to Westminster.

The passenger can complete this journey utilizing only the District line or only the Circle line, but he may also transfer between the two lines as many as four times, once at each intermediate station. The two journeys are identical from the perspective of the basic minimum fare path method. Preferably, when multiple paths producing the minimum fare exist, the algorithm would select the one with minimum duration. If multiple paths meeting this secondary criterion exist, then the algorithm should select from the subset the path with the minimum number of transfers.

A slight adjustment to the minimum fare path method makes this tie breaking possible. Under the minimum fare path method, the algorithm considers only the incremental fare cost assigned to each arc. Thus, assigning any additional cost, no matter how small, to an arc makes it a less desirable addition to the journey path. If this cost is small enough relative to the zonal fares, then the algorithm will still produce the cheapest possible fare. If the additional costs are scaled correctly, the algorithm will also select the path meeting the secondary cost criteria. Defining the composite cost of each arc as follows produces the minimum fare path with the shortest duration:

1. Scale up the fare increment by a large integer. For example, transform a fare of 250 pence to 250,000,000 pence by multiplying the base fare by 1,000,000. Set the composite cost of the arc equal to this scaled value.

2. Scale up the duration of travel on this arc by another large, yet smaller, integer and add this value to the arc's composite cost. For example, if the arc being added has an actual incremental fare of 250 pence and duration of eight minutes, scaling the duration by 1,000 will produce a composite cost of 250,008,000.

3. Add one unit to the composite cost if a transfer is required to eliminate needless transfers. For example, if utilization of the arc in (1) and (2) required a transfer, then the composite cost for this arc could be treated as 250,008,001.

Assuming the scalar values discussed above, minimizing duration will necessarily dominate minimization of the number of transfers, unless a journey requires more than 999 transfers. For every transfer in excess of 999 transfers, the algorithm will be unable to distinguish between a transfer and an additional minute of duration. Similarly, as long as total journey duration is less than 999 minutes, minimizing the journey fare will necessarily dominate the minimization of duration. For every minute of duration in excess of 999 minutes, the algorithm will fail to distinguish between duration and an additional one currency unit increase in fare. It is unlikely that a journey will require more than 999 transfers or last in excess of 999 minutes. However, if duration is defined in units of seconds rather than

minutes, the composite cost (with scalars as defined above) fails at journey duration of 16.7 minutes. In this case, of course, different scale factors should be chosen.

Adjusting the scalar values accordingly allows the algorithm to accommodate any unit definition. Thus, this approach does not adversely affect the selection of the truly minimum fare path and, in the event of a tie, forces the algorithm to select the path meeting the secondary and tertiary criteria. This approach also does not affect tracking of the path variables; the fare, duration, number of transfers, and distance can all be retrieved directly from the array in which each is stored.

(b) Tie breaking under minimum duration path

The minimum duration path method is likely to produce a unique solution much more frequently than the minimum fare path. Additionally, if accurate travel and transfer time data are entered, the algorithm already captures the cost of transfers. Still a case may arise where two or more paths produce the same minimum journey duration, but visit different zones and thus produce different fares. In this case, the algorithm should select the minimum duration path with the lowest fare.

The primary consideration under the minimum duration path method is the duration associated with each arc, incorporating incremental fare as an additional selection criterion must not interfere with duration tracking. Defining the composite cost of each arc as follows produces the minimum duration path with the least number of transfers and smallest fare:

1. Scale up the duration of travel increment for a given arc by a large integer. For example, transform a duration of eight minutes into 8,000,000 minutes by multiplying the arc's duration by 1,000,000. Set the composite cost of the arc equal to this scaled value.

2. Add one scaled unit to the composite cost if a transfer is required to eliminate needless transfers. For example, if utilization of the arc in (1) required a transfer, then the composite cost for this arc could be treated as 8,001,000 by adding 1,000 to the composite cost produced in (1).

3. Add the incremental fare associated with use of an arc to the composite cost produced in (2). For example, if utilizing an arc with a travel time of eight minutes requires a transfer and has an incremental fare of 150 pence, then the composite cost for this arc would be 8,001,150.

Assuming the scalar values discussed above, minimizing duration will necessarily dominate minimization of the number of transfers, unless a journey requires more than 999 transfers. For every

transfer in excess of 999 transfers, the algorithm will be unable to distinguish between a transfer and an additional minute of duration. Similarly, as long as total journey fare is less than 999, minimizing the number of transfers will necessarily dominate minimization of journey fare. For every one unit of currency in excess of 999, the algorithm will fail to distinguish between a transfer and an increase in fare. If journey fares of greater than 999 pence are possible (based on the fare structure), then the scalar values must be adjusted to prevent improper path selection.

The values of the scalars used for each component of the composite cost should reflect, at a minimum, the maximum possible value for each component. If fares in excess of 999 pence are reasonably expected—if at least one zone-to-zone fare is greater than 999 pence—then both scalar values need to be adjusted accordingly. Assume the maximum fare defined in the zonal fare table is 15,000 pence. The scalar used to include transfers in the composite cost must be greater than 15,000 to ensure the algorithm accurately distinguishes between transfers and fare. Similarly, if the scalar used for transfers is increased, so must the scalar for duration to ensure the algorithm accurately distinguishes between transfers and duration.

(c) Tie breaking under minimum transfer path

The minimum transfer path method will often have non-unique solutions. The most logical secondary path selection criterion for tie breaking under the minimum transfer path is duration. Fare minimization is most appropriately considered a tertiary tie-breaking criterion. Defining the composite cost of each arc as follows produces the minimum transfer path with the smallest duration and smallest fare:

1. Add one scaled unit to the composite cost if a transfer is required to eliminate needless transfers. For example, if utilization of an arc requires a transfer, then the composite cost for this arc could be treated as 1,000,000 by adding 1,000,000 to the composite cost for each transfer.

2. Scale up the duration of travel for the arc by a large integer and add it to the composite cost. For example, if the arc in (1) carries a duration of eight minutes, the composite cost could be 1,008,000.

3. Add the incremental fare associated with use of an arc to the composite cost produced in (2). For example, if utilization of an arc that carries a duration of eight minutes requires a transfer and incremental fare of 150 pence, then the composite cost for this arc would be 1,008,150.

Assuming the scalar values discussed above, minimizing the number of transfers will necessarily dominate minimization of duration, unless total journey duration exceeds 999 minutes. For every minute of duration in excess of 999, the algorithm will be unable to distinguish between a transfer and additional duration. Similarly, as long as total journey fare is less than 999, minimizing the journey duration will necessarily dominate minimization of journey fare. For every one unit of currency in excess of 999, the algorithm will fail to distinguish between duration and an increase in fare. If journey fares of greater than 999 pence are possible (based on the fare structure), then the scalar values must be adjusted to prevent improper path selection as described in 4.2.5(b).

4.2.6 Intermediate validation

In the physical network, an intermediate validation occurs when a passenger taps his or her payment device on a validator at some point during the journey. This action provides the fare calculation system more information about the passenger's actual behavior behind the gate line, and thus the algorithm can more accurately represent the actual journey. Intermediate validation occurs today in London in a variety of cases including out-of-station interchanges, transitions from the Underground to the DLR, or transitions from the Underground to tram services. Future fare policy may allow TfL to adjust a given journey's fare based on a passenger's intermediate validation behavior.

Representing intermediate validation in the directed graph model of the London transit network requires a multi-step approach. Intermediate validation is, in many ways, equivalent to forcing the algorithm to choose a path from origin to destination that passes through a given (intermediate validation) node. However, the shortest path algorithm produces the path and fare from a given origin node to every other reachable node in the network based on the selected criteria. There is no direct way to force the algorithm to include a given node in the path from origin to destination. Moreover, to ensure accurate transfer representations, the network representation must treat a validator as a terminal node. Therefore, once a validator node is added to a path, the path necessarily terminates. If the path terminates at a validator node, it does not represent a full journey. Instead, representing a single journey that includes intermediate validation requires aggregating a series of separate paths.

The first path segment of a journey with intermediate validation can be represented as a sub journey from the origin station's entrance node to the validator node at the intermediate validator station. The algorithm calculates the shortest path between these two nodes based on the chosen path selection method. Once the path is calculated, the algorithm temporarily stores the validator nodes' relevant tracked variables (all tracked variables except composite cost) for aggregation and future path selection.

The next segment of the journey originates at the last station-service sub node visited in the previous path segment and terminates at the exit node of the destination station. The origin node for the second segment is easily determined because both the intermediate validator station and the last service used are known from the temporarily stored variables—the ID of every node associated with a station is stored in a two-dimensional with the indices corresponding to station ID and service ID.

The innermost and outermost zones visited, as well as the last service used to arrive at the validator node in the first sub journey, are associated with the new root node. This ensures the algorithm will select the path for this segment of the journey correctly based on the first journey segment; the zones visited influence fare calculation, and the last service used is important for tracking transfers and in determining the appropriate origin node of the second journey segment. Distance, duration, the number of transfers, the array of predecessor nodes, and the array of last service used to reach each node for the shortest path solution of the first journey segment are each important in describing the complete journey, but none of these affects the algorithm's decisions in subsequent journey segments.

The algorithm then selects the shortest path for this journey segment based on the chosen path selection method. If this second journey segment completes the physical journey (if there is only one intermediate validation), the algorithm terminates and reports the aggregate journey information. The algorithm adds the duration, distance, and number of transfers for this segment to the values stored from the first segment. The arrays of predecessor nodes and last service used to access each node that describe the first journey segment are kept separate from the arrays for the second segment so that each segment can be easily identified when reported. Composite cost is not important for reporting purposes. The remaining tracked variables can be retrieved directly from the values determined by the shortest path algorithm for the second segment. These variables include innermost and outermost zones visited, as well as the journey fare.

If the journey includes additional intermediate validations, each segment is defined in a similar way. A journey segment between two intermediate validation stations has the last station-service sub node visited in the previous segment as its origin and the validator node of the next intermediate validation station as its destination. This process can be repeated as many times as desired until the exit node of the journey's destination station is reached.

4.3 Tracking variables

The algorithm finds the shortest path from the root node to every other node in the network, not just a selected destination. As described in section 2, the algorithm considers each arc out of the node on

the front of the candidate list. For this discussion, the node on the front of the candidate list is the *active node* and the node connected to the active node via the arc under consideration is the *considered node.*

The algorithm tracks variables describing the shortest path for decision and reporting purposes. The algorithm stores these labels for each node to describe the shortest path from the root to each node. Each label is stored in one more arrays. The algorithm updates the information stored for a node whenever it identifies a path shorter than the stored path.

4.3.1 Tracking visited zones

All LUL fares are calculated based on the innermost and outermost zones visited on a given journey. Without tracking the innermost and outermost zones visited, the algorithm would not be able to produce an accurate journey fare. Thus tracking the innermost and outermost zones visited along a path is essential not only in the minimum fare path method, but also in every other path selection method. As can be seen on the standard tube in Figure 4.1, central London stations fall into the Zone 1 category, the innermost and most expensive zone. Eight additional zones surround Zone 1.

For each of the path selection methods, the algorithm maintains two one-dimensional arrays to track the innermost and outermost zones visited; these are additional 'labels' for each node, similar to cost and predecessor variables at the core of the shortest path algorithm. The index of each array is the node ID. The values stored in the arrays are the respective innermost and outermost zones visited in traveling along the shortest path from the origin node to the given node. The algorithm uses these stored values to determine the appropriate fare from a lookup table of the basic zonal fare structure, which is stored as a multi-dimensional array for easy access.

The algorithm updates the zone-visited labels every time a path shorter than the previously assumed path visits a given node. Shorter, again, is defined according to the path selection method being employed. The updating process calculates and temporarily stores the innermost and outermost zones visited as separate variables during each progressive step in the path selection—the algorithm does not update the stored label while considering a node.

The algorithm always starts at the user-defined origin node, which is a station entrance node in most cases, but may also be a station-service sub node in the case of a journey requiring intermediate validation. Once the user selects an origin node, the algorithm updates the origin node's zone-visited labels. The values assigned to the origin node's zone-visited labels are dependent on the origin node's type:

- If the origin node is *not* the first node in a journey, the algorithm assigns the zone-visited labels produced in the previous journey segment.
- If the origin node is the first node in a journey (i.e., a station entrance node), and assuming the origin node is not associated with a boundary station, the algorithm assigns the origin node's zone to both zone-visited labels. Boundary stations require special consideration as described later in this section.

The algorithm can only consider a node if it is reachable via a single arc from the current active node. Each time the algorithm considers a connected node, it compares the zone-visited labels of the *active node* with the zone of the considered node. The temporary inner and outer zone variables are assigned the values of the active node's zone-visited labels. If the zone of the considered node is larger than active node's outermost zone visited, the temporary outer zone variable is updated with the considered node's zone. Similarly, if the considered node's zone is less than the active node's innermost zone visited, then the temporary inner zone variable is updated with the considered node's zone. If the considered node's zone meets neither of these conditions, the temporary variables reflect the active node's zone-visited labels.

If the algorithm has found a shorter path to the considered node, it updates the considered node's stored zone-visited labels. The considered node's innermost zone visited label is replaced with the value of the temporary inner zone variable. Likewise, the outermost zone visited label is updated with the value of the temporary outer zone variable. Even though the algorithm has identified a shorter path to the considered node, the new values of the zone-visited labels may not differ from the previously stored values. However, the algorithm has necessarily altered at least one of the considered node's labels; otherwise, it would not have identified the current path as an improvement.

(a) Tracking zones for boundary stations

An added layer of complexity in London's zonal fare structure is that of boundary station, which is a station located on the border between two zones. Approximately 30 LUL stations are boundary stations. If a boundary station is included in a journey path, the station coincides with whichever of the two zones produces the lower fare.

Recall that travel in an inner zone is always at least as expensive as travel in an outer zone, and the algorithm only considers a node if it is connected to the active node. Selecting the boundary station's outer zone produces the cheapest fare unless the path visits a station in a lower zone at some other point

along the journey. If the journey can be completed without visiting a zone greater than the boundary station's inner zone, selecting the inner zone will produce the cheapest fare.

The algorithm selects the proper zone by comparing the two possible zones with the zone-visited labels for the *active* node. If the active node's innermost zone visited is greater than or equal to the larger of the two boundary zones, the algorithm assumes the station is in the outer of the two zones. If the active node's outermost zone visited is less than or equal to the smaller of the two boundary zones, the algorithm assumes the station is in the inner zone. When the algorithm identifies a shorter path to a node associated with a boundary station, it updates the stored labels for the node accordingly. Visiting subsequent nodes and zones may result in changes to a journey's innermost and outermost zones visited, but the algorithm will not reconsider the zone selection for this boundary station node unless it is revisited.

When a journey originates at a normal station, the innermost and outermost zones visited for the origin node will each be the single zone associated with the origin node. If a path originates at a boundary station, the zone-visited labels for the origin node must be determined differently. The algorithm assigns the boundary station's outer zone to the origin node's innermost zone-visited label. The origin node's outermost-zone-visited label is assigned the boundary station's inner zone. This seems counterintuitive, but is necessary for the algorithm to select the zone producing the cheaper fare.

Assigning the initial values in this way ensures that the rules described in section 4.3.1 will not update the innermost and outermost zones visited until a non-boundary node (or a boundary node with different zones) is considered. If the algorithm considers multiple boundary nodes in an uninterrupted

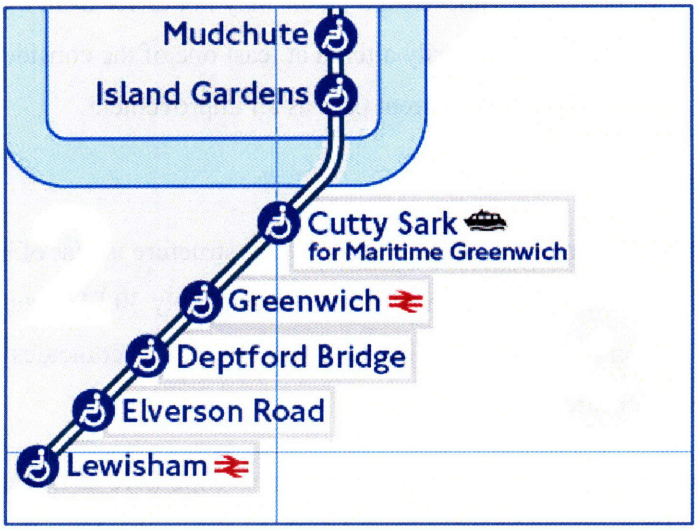


Figure 4.14: Journeys that visit only boundary stations (indicated by a box around the station name) require special consideration for fare calculation.

sequence stemming from the origin boundary node, the zone-visited labels for each of these nodes will match those of the origin node. Figure 4.14 illustrates an area in the TfL network where such a sequence occurs. If a journey originates at Lewisham, for example, the algorithm assigns zone 3 (the outer boundary zone) to the origin node's innermost-zone-visited label and zone 2 (the inner boundary zone) to the origin node's outermost-zone-visited label. As the algorithm visits each successive station up to and including Cutty Sark, it assigns these same values to each node's zone-visited labels.

The uninterrupted sequence of boundary stations is the only case that results in the innermost-zone-visited label exceeding the outermost-zone-visited label. The incremental fare calculation can therefore identify and process this special case as described in section 4.3.2 (a).

When the algorithm visits the first non-boundary node in the journey, the rules of section 4.3.1 assign values to the temporary zone-visited variables that minimize the fare. If the above journey continues beyond Cutty Sark to Island Gardens, for example, the algorithm assigns zone 2 as Island Gardens' innermost-zone-visited label and zone 3 as its outermost-zone-visited label. For this node, and every subsequent node, the outermost zone visited variable is greater than or equal to the innermost zone visited variable. Zonal tracking then continues as normal.

4.3.2 Tracking fares

LUL fares are determined based on the innermost and outermost zones visited on a given journey. TfL defines the basic pay-as-you-go fare structure for every possible zone-to-zone combination. TfL's zone-to-zone fare structure effective 1 January 2008 includes nine zones, or 45 unique combinations, for each of five fare types (see Appendix A). Representing multiple fare types, including concessionary fares and peak versus off-peak travel, is accomplished by defining a basic fare structure for each fare type. Loading the fare structure into a three-dimensional array (the first two dimensions indicating the inner and outer zone and the third dimension indicating the fare type) allows the algorithm to determine the applicable fare for any given pair of zones, under any defined fare type. The algorithm also maintains a one-dimensional array of the calculated journey fare to reach each node via the shortest path to that node; each entry in this array is an additional node label similar to the previously discussed labels.

When the algorithm initializes, it sets every node's journey fare label to a large integer to represent an infinite fare. This allows the algorithm to determine whether it has previously visited a given node. Assigning the fare label for the origin node is dependent on the type of node:

- If the origin node is the first node in the journey, its fare label is assigned a value of zero. This does not imply that a user cannot be charged an entry fee. Rather, it ensures correct calculation of the incremental fare to each node directly accessible from the origin.

- If the origin node is the start of any journey segment other than the first, the fare label is replaced with the fare produced to reach this node in the previous segment.

Regardless of the path selection method, the algorithm calculates the journey fare increment each time a node is considered based on a combination of the temporary zone-visited variables and the zone-visited labels stored for the *active* node. Calculating fares incrementally is necessary for the minimum fare path method and provides support for variations in the fare structure. Generally, the fare increment for any considered arc and node does not depend on the fare label of the active node. However, the calculation of the considered node's journey fare label does depend on the active node's fare label.

TfL's current fare policy makes basic fare calculation strictly a function of the innermost and outermost zones visited. However, there are a few instances where this rule may be relaxed even while maintaining the present fare structure. One example is the inclusion of a discounted fare for the portion of a journey occurring on a service suffering significant delay or disruption. A journey utilizing only the disrupted service would not require any special treatment other than an adjusted zonal fare structure; fare calculations based only on the temporary zone visited variables would remain accurate. However, if portions of the journey occur on unaffected services, it may be desirable to charge the full fare for travel occurring before the disrupted service is encountered and for travel occurring after the disrupted portion. Calculating fares incrementally supports this approach.

An arc's fare increment is the amount by which the journey fare increases when the algorithm utilizes the arc to reach a node. Each time a node is considered, the algorithm calculates the arc's fare increment as follows:

1. Look up the fare for the active node's innermost and outermost zones visited from the base fare structure. This lookup fare may differ from the active node's stored journey fare label; the stored fare may be less than the lookup fare if discounts have been applied, or it may be greater if a surcharge has been imposed (perhaps for congestion or express service). Do not apply any discounts or adjustments to any node other than the root. The root node's lookup fare is always adjusted to zero.

2. Look up the fare for the considered node's temporary innermost and outermost zone visited variables. Adjust this lookup fare by any service or node specific discounts applicable to the arc and node under consideration.

3. Subtract (1) from (2).

4. The difference produced in (3) will be positive in all cases except when transitioning from a normal journey segment to a discounted segment. If (3) produces a positive value, the fare increment is that value. If (3) produces a negative value, the fare increment is zero.

Restricting the fare increment to a non-negative value ensures the algorithm will not alter the portion of the journey fare incurred for travel preceding the discounted segment. Determining the fare increment based on (1) rather than the active node's journey fare label ensures travel in the discounted segment remains discounted, and surcharged segments remain surcharged, even if normal segments follow. If the algorithm determined the fare increment using the active node's fare label instead of the lookup value in (1), the transition arc from discounted service to normal service would carry a fare increment including the difference between the normal and discounted fares. A journey including travel after this transition would not reflect any of the previous discounts, or it may carry only a portion of any imposed surcharges.

The algorithm sums the fare increment and the active node's fare label to produce the aggregate journey fare. This aggregate value is stored as a temporary variable while the algorithm considers updating the path to the considered node. If the algorithm identifies a shorter path, it replaces the considered node's journey fare label with the temporary fare variable.

Consider the five-station, eight-node, journey path example in Figure 4.15. Assume station 9 is zone 1, stations 10 and 11 are in zone 2, and stations 12 and 13 are in zone 3. The zone-1-only fare is £1.50, the zone-1-to-zone-2 fare is £2.00, and the zone-1-to-zone-3 fare is £2.50. Furthermore, assume TfL has reduced the fare for travel on service B between stations 11 and 12 by 50% of the normal fare to compensate passengers for delays.

Table 4.1 summarizes the results of the fare calculation for this sample journey. Only the first two arcs in this journey have positive fare increments, the rest are zero. When the algorithm considers the discounted arc between 11b and 12b, it adjusts 12b's lookup fare by the 50% discount to £1.25. Node 11b's lookup fare (as the active node) is £2.00. The difference between these lookup fares is negative, so the algorithm assigns the arc zero fare increment. As the journey continues beyond the discounted section

and node 13b becomes the considered node, node 12b's lookup fare (as the active node) is the non-discounted zone-1-to-zone-3 fare, or £2.50. The difference between the lookup fares for 13b and 12b is zero, the travel through the zone-1-to-zone-3 transition remains discounted. Thus, the discounted journey fare is £2.00 rather than the £2.50 full fare.

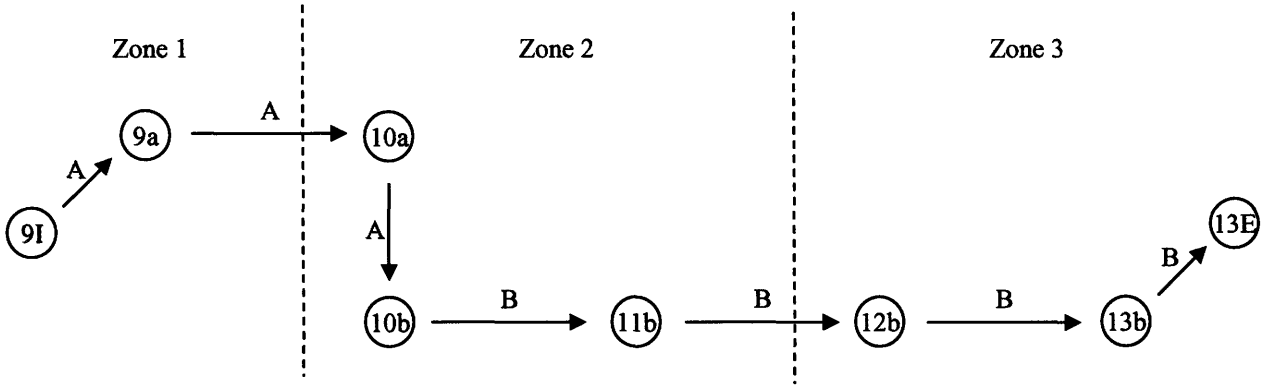


Figure 4.15: This eight-node sample journey path spans three zones. The fare for the portion of the journey between nodes 11b and 12b has been reduced by 50% to compensate passengers for delays.

Table 4.1: The algorithm calculates journey-fare labels using incremental arc costs. An arc's cost is the greater of zero and the difference between the considered and active node's lookup fares. A given node's lookup fare is only adjusted when that node is the considered node, never when it is the active node.

Node ID	Zone-visited labels		Lookup fare (£)		Previous arc's increment (£)	Journey-fare label (£)
	Innermost	Outermost	Active	Considered		
9I	1	1	0	0	-	0
9a	1	1	1.50	1.50	1.50	1.50
10a	1	2	2.00	2.00	0.50	2.00
10b	1	2	2.00	2.00	0	2.00
11b	1	2	2.00	2.00	0	2.00
12b	1	3	2.50	1.25	0	2.00
13b	1	3	2.50	2.50	0	2.00
13E	1	3	2.50	2.50	0	2.00

(a) Tracking fares for boundary stations

When a journey originates at a node associated with a boundary station, the value of the innermost zone visited label is greater than the value of the outermost zone visited label. The fare stored for the origin node is still zero as in the normal case. However, the fare increment of any arc out of the origin node requires special treatment.

When the active node is the origin node (identified by a journey fare of zero) and a boundary station (identified by inverted zone-visited labels), the fare increment for the considered arc depends on

whether or not the considered node is also a boundary node. If the considered node is a boundary node with the same zones as the origin node, the fare increment is the single zone fare that applies to travel in only the outermost zone of the boundary nodes. As described in section 4.3.1 (a), the boundary origin node's outermost zone is actually stored as the origin's innermost zone visited label.

If the considered node is not a boundary node, the temporary outermost zone visited variable will be at least as large as the temporary innermost zone visited variable. The fare increment is the fare that applies to a journey between the temporary zone-visited variables. This case will not likely occur under the network representation discussed in section 4 because the origin node of a journey is always a station ingress node. The only arcs out of a station ingress node lead to nodes associated with the same station. Therefore, the first node considered after an origin node at a boundary station will always be another boundary node. These special cases do not usually apply to journey segments after the first segment because the zone-visited labels will not be inverted (unless all of the previous segments visited only boundary nodes of the same zones).

The second arc utilized in a journey originating at a boundary node also requires special consideration. The first two nodes visited were boundary nodes and therefore the current active node's stored zone-visited labels remain inverted. If the considered node is also a boundary node in the same zones as the active node, the fare increment is zero. The fare increment is also zero for any subsequent boundary nodes in the same zones.

If the considered node is not a boundary node, the temporary zone-visited variables will reflect the minimum fare zone selection as described in section 4.3.1. The algorithm calculates the fare increment as follows:

1. Look up the single zone fare for the active node's innermost zone visited, which is the larger of the boundary station's two zones. If the root node is the origin, adjust this lookup fare to zero.

2. Look up the fare for the temporary innermost and outermost zone visited variables for the node under consideration. Adjust this lookup fare by any service or node specific discounts applicable to the arc and node being considered.

3. Subtract (1) from (2).

4. If (3) produces a positive value, the fare increment is that value. If (3) produces a negative value, the fare increment is zero.

In each case, the algorithm calculates the journey fare for the considered node by summing the fare increment and the active node's fare label.

4.3.3 Tracking transfers and last service

Tracking the number of transfers between services or between different branches of a given service is essential for the operation of the minimum transfer path method, and is also necessary for reporting path statistics under any of the path selection methods. The algorithm tracks the total number of transfers required to reach each node as another node label and stores these values in a one-dimensional array. Determining whether a transfer is necessary for travel from the active node to a considered node requires knowledge of both the considered arc's service and the last service used to reach the active node. This algorithm accomplishes this by tracking the last service used to reach each node as another node label and storing these labels in a one-dimensional array.

When the algorithm initializes, it assigns each node's previous-service-used label a value of -1 to indicate the node has not been visited. Similarly, the algorithm assigns a large integer value to every node's number-of-transfers label to represent infinite transfer cost. The algorithm updates each of these labels for the origin node based on the node type:

- If the origin node is the first node in the journey (i.e., a station entrance node), the origin node's number-of-transfers label is updated with a value of zero. The origin node's last-service-used label is not altered from the value of –1. Assigning the origin node a previous service would result in an inaccurate assessment of transfers required within the origin station.

- If the origin node is station-service sub node, indicating it is the start of a journey segment other than the first, the algorithm updates both labels to reflect the node's labels produced in the previous journey segment.

Each time the algorithm considers an arc and node combination, it compares the arc's service ID with the stored last service ID for the active node. If these values are identical, or if the active node's last-service-used label is –1, the algorithm assumes no transfer is necessary. If these values are different and the active node's last-service-used label is greater than –1, then the algorithm assigns a transfer increment of one unit. The algorithm adds the active node's number-of-transfers label to the transfer increment and holds this value as a temporary variable while considering updating the path.

If the algorithm identifies a shorter path, this summed value is stored as the considered node's number-of-transfers label. The algorithm then updates the considered node's last-service-used label with the considered arc's service ID. None of the active node's labels are altered.

4.3.4 Tracking duration

Tracking duration is essential for the minimum duration path method and plays a key role in each of the other path selection methods. The algorithm tracks the duration required to reach each node via the shortest path as an additional node label and stores these values in a one-dimensional array.

When the algorithm initializes, it assigns a large integer value to each node's duration label to represent infinite duration. The algorithm updates the origin node's duration label with a value of zero to indicate no travel has occurred to this point, regardless of the node type.

Each time a node is considered, the duration increment is determined by retrieving the considered arc's associated duration. The algorithm adds the active node's duration label to this increment and holds the summed value as a temporary variable while considering whether to update the path. If the algorithm identifies a shorter path, it updates the considered node's duration label with the value of the temporary variable.

4.3.5 Tracking distance

Tracking distance is essential for the minimum duration path method and plays a key role in each of the other path selection methods. The algorithm tracks the distance required to reach each node via the shortest path as an additional node and stores these values in a one-dimensional array.

When the algorithm initializes, it assigns each node's distance label a large integer value to represent infinite distance. The algorithm updates the origin node's distance label with a value of zero to indicate no travel has occurred to this point, regardless of the node type.

Each time the algorithm considers an arc and node combination, it retrieves the arc's distance increment. The algorithm adds the active node's distance label to this increment. It holds the summed value as a temporary variable while considering whether to update the path to the considered node. If the algorithm identifies a shorter path, it assigns the value of the temporary variable to the considered node's distance label.

4.3.6 Tracking predecessor nodes

Tracking predecessor nodes is important for the basic function of the algorithm and for reporting the actual node-by-node description of the selected path. The node ID of every node's predecessor is stored as an additional node label in a one-dimensional array. This allows recovery of the shortest path for any node by tracing each node's predecessor starting from the destination node.

When the algorithm initializes, it assigns a value of -1 to every node's predecessor label. The origin node's predecessor label is not altered from –1 because assigning any non-negative value would indicate a node precedes the origin, which is impossible by definition of the origin node.

No action is necessary for tracking the predecessor node while the algorithm considers a node. However, if the algorithm indentifies a shorter path, it updates the considered node's predecessor label with the active node's ID.

4.3.7 Tracking composite cost

Composite cost is a derived value used for tie breaking among multiple feasible routes. Section 4.2.5 discusses the components and calculation of the composite cost. The algorithm tracks the composite cost to reach each node as an additional node label and stores these values in a one-dimensional array.

When initialized, the algorithm assigns every node's composite-cost label a large integer value to represent infinite composite cost. The algorithm updates the origin node's composite-cost label with a value of zero to indicate travel is costless to this point.

The algorithm calculates the incremental composite cost each time a node is considered based on the rules of section 4.2.5. The algorithm adds the active node's composite-cost label to the incremental composite cost and holds this value as a temporary variable. If the algorithm identifies a shorter path, it updates the considered node's composite-cost label with this summed value.

The composite cost is the only variable considered by the algorithm under each path choice method. This allows tie breaking to occur efficiently during real-time path selection. Tracking of all other variables is necessary for path selection and reporting because the algorithm computes the composite cost from these variables.

4.4 Retrieving path information

When the shortest path algorithm terminates, the stored labels for each tracked data item correspond to the shortest path from the root node to any given node. The algorithm has visited every node that is reachable via some path originating at the root node. Tracked data items for disconnected nodes—those nodes that are not accessible from the root node—will therefore remain in their initial states of "infinite" cost and can be easily distinguished from visited nodes.

Providing support for intermediate validation taps requires appropriate tracking of aggregate journey data, as described in section 4.2.6. Each time the algorithm runs (for each journey segment) it must retrieve the relevant data for aggregation. Most of the tracked path information is recovered directly from the node labels stored in each respective array—recall the array's index corresponds to node IDs. Each of the following tracked data items can either be recovered directly from an array, or recovered from an array and aggregated with the result from the previous journey segment:

- calculated fare,
- innermost zone visited,
- outermost zone visited,
- total number of transfers,
- total duration, and
- total distance.

When a path begins at a boundary station, the labels for the innermost and outermost zones visited are set equal to the outer and inner zones of the boundary station, respectively (see section 4.3.1 (a) for more detail). The innermost zone visited value will always be correct:

- If the path includes only boundary stations, each node's innermost-zone-visited label will be the outer zone of the boundary stations.

- Otherwise, the algorithm updates the innermost zone visited label as appropriate.

However, the values stored for the outermost zone visited may be inaccurate for paths including only boundary nodes. If a path includes only boundary nodes, the true outermost zone visited is the outer zone of the boundary stations. To ensure proper zone tracking, the algorithm sets the outermost-zone-visited label equal to the boundary station's inner zone for every node in the path. In this case, the algorithm cannot retrieve the outermost zone visited directly from the outermost-zone-visited label. For any node, if

the innermost zone visited is greater than the outermost zone visited, the algorithm retrieves the outermost-zone-visited label in place of the innermost zone visited.

The actual service sequence and node-to-node path description is produced by combining the tracked predecessor nodes and previous service used data items. Once the user selects a destination station, the final node in the journey is the exit node associated with that station. The path from root node to destination exit node can then be determined in reverse as follows:

1. The ID of the final service utilized to reach the exit node can be determined from the exit node's previous-service-used label.

2. The exit node's predecessor label indicates which node immediately precedes the exit node in the shortest path.

3. The ID of the last service used to reach the node preceding the exit node is retrieved from the predecessor node's last-service-used label. Under the graphical representation discussed in section 4.1, this service ID will necessarily match the exit node's service ID.

4. The ID of the third-to-last node visited in the shortest path is found in the array of predecessor labels at the index corresponding to the second-to-last node; this index is the value discussed in (2).

5. The ID of the last service used to reach the third-to-last node is retrieved from this node's last-service-used label. If this service ID differs from the ID of the last service used for the next node (the second-to-last node), then a service transfer has occurred. This transfer has necessarily occurred within the station associated with the third-to-last node.

6. Repeating (4) and (5) until reaching the root node provides the complete description of the node-to-node, transfer, and service path in reverse order.

To represent the node-to-node path in order from root to destination:

1. Store each node ID in a temporary, expandable array as it is recovered from the array of predecessor nodes.

2. Create a new array of fixed size equal to the final size of the array in (1).

3. Place the last entry of the array in (1) in the first slot of the new array in (2).

4. Fill each consecutive entry of the new array in (2) using the value from the highest index yet to be transferred.

4.5 Staff and passenger information device

Using the algorithm to assist with manual fare assignment or to provide path information to staff or passengers will require a graphical user interface (GUI). Figure 4.16 displays a screen capture of the GUI developed for this research. Specific documentation for the developed GUI is available in a memo dated May 8, 2008 with subject "TfL fare calculator GUI."

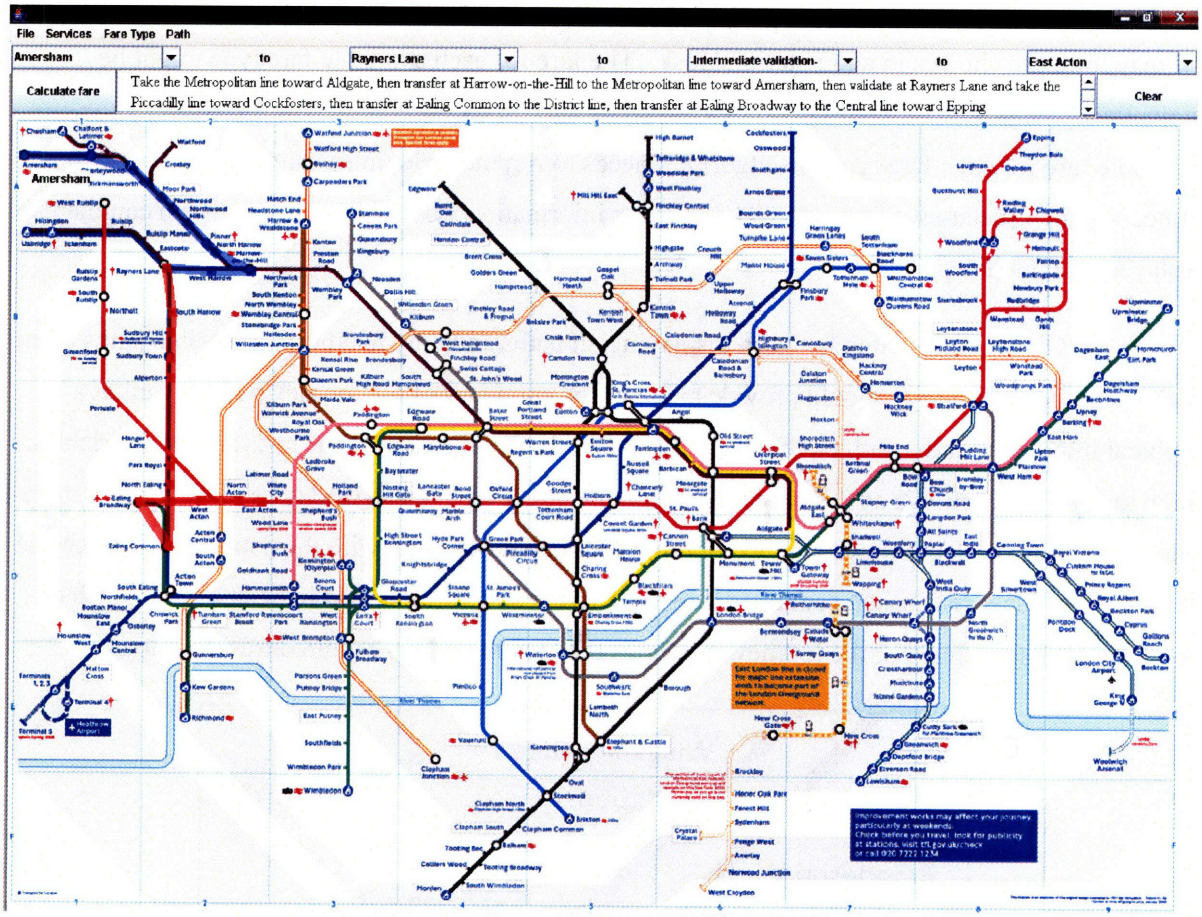


Figure 4.16: A graphical user interface was developed as part of this research to display generated path and fare information for TfL staff and passengers.

The GUI needs to be simple and intuitive. The user must easily be able to indicate the desired origin, destination, intermediate validations, fare type, and path choice method. The GUI must then present as much relevant data as possible without overwhelming the user.

Using the iconic Tube map as the central focus of the interface allows users to quickly indicate their desired origin and destination stations. An in-station kiosk might use a large touch screen to provide path and fare information after a passenger touches his or her desired destination. A similar touch or click interface might be employed in a desktop version of the information device. The map coordinates of each station can be stored as an attribute in the stations table of the database. These coordinates will be associated with the specific map image chosen. Drop-down boxes containing all enabled stations could also be used to allow users to select the desired origin, destination, and any intermediate validation stations.

Each of the path selection methods should be available as an option for planning purposes, but TfL may limit the choices in an in-station kiosk. The fare for each available fare type might be automatically displayed or the user could be asked to select the desired fare type. To display the applicable fare for each fare type, the algorithm needs to run multiple times using the fare structure defined for each successive fare type. The relevant information stored following each run can then be displayed in aggregate.

The GUI should switch between displaying all paths and only handicap accessible paths at the user's indication. The GUI should provide a text-based summary of the journey path as well as a graphical indication of the path on the map. The algorithm tracks each of the relevant path variables, which the GUI can retrieve as described in section 4.4. Section 4.4 also discusses retrieving the node-by-node description of the journey path. Combining this path description with the data in the last service used array produces text-based path instructions. These instructions should indicate what branch of service is used in each leg of the journey as well as the location of any required transfers and/or

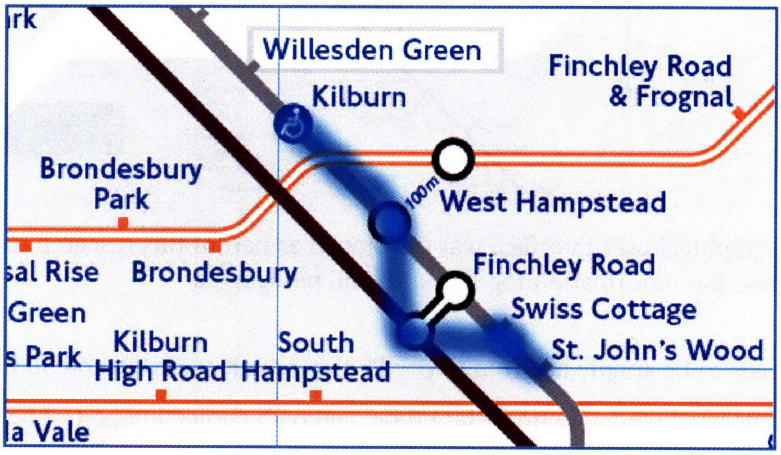


Figure 4.17: The path drawn using default station coordinates is misaligned as it passes through Finchley Road Station because the map represents the same station at different coordinates for the Jubilee and Metropolitan lines.

intermediate validations.

The GUI can use the node-by-node path description to draw the journey path on the map. The drawn path is a series of lines, each starting at the coordinates of a node in the journey path and ending at the coordinates of the next node in the path. If the map represents a station in two locations, the path drawn may appear misaligned such as the case of Finchley Road in Figure 4.17. This issue arises because all of the station-service sub nodes associated with a given station share the same coordinates. Overriding the sub node's default coordinates defined by the station attributes with coordinates defined for the specific service link helps the path drawn match the map image.

5 Database

The algorithm reads all of the required network and fare structure information from a database. Using a database helps ensure referential integrity and allows quick alteration of any component. Figure 5.1 below shows an entity relationship diagram of the database structure required to support the path selection and fare calculation algorithm. A brief description of the entities and their attributes follows.

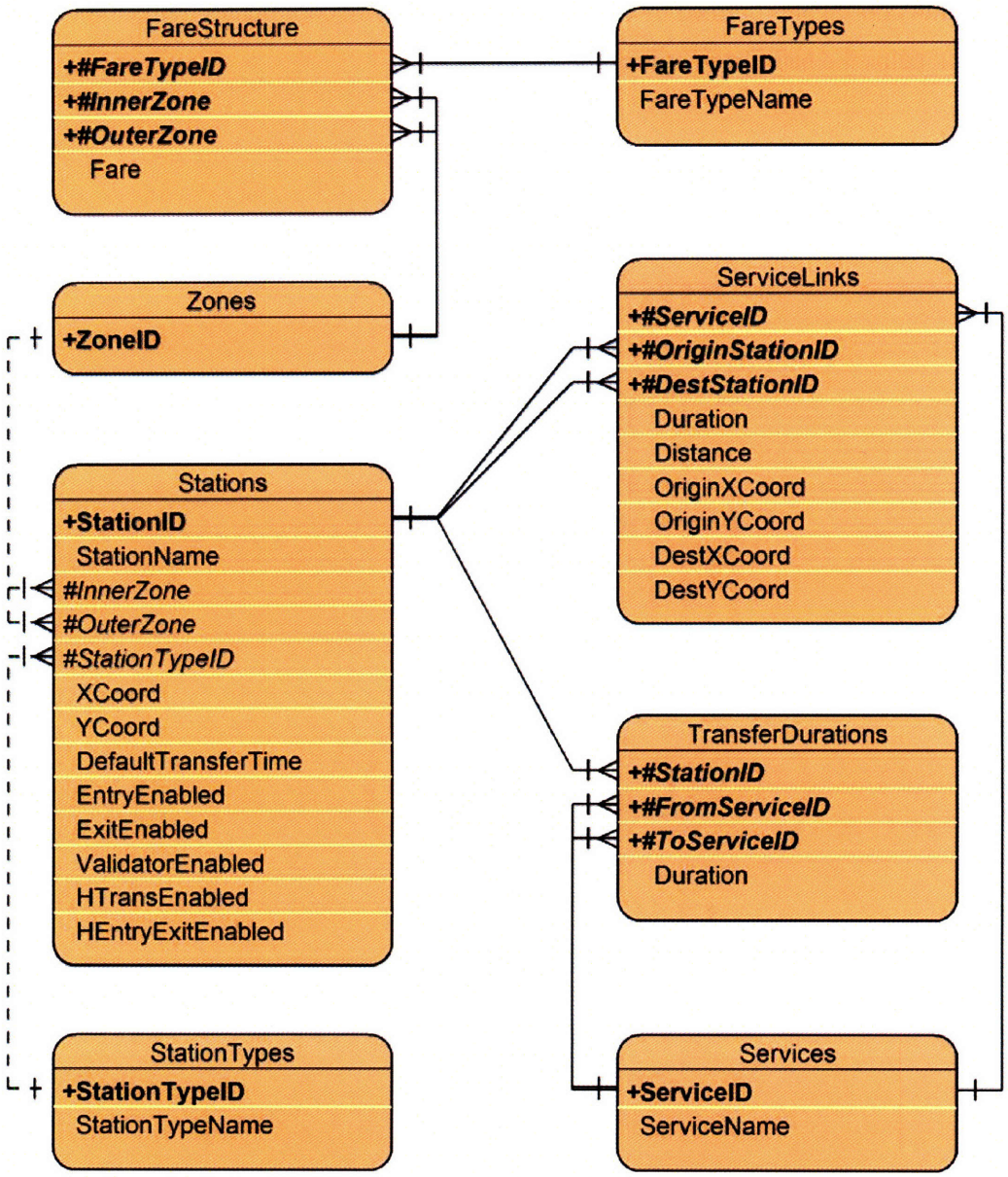


Figure 5.1: This entity relationship diagram shows the basic database structure required.

5.1 FareStructure table

The FareStructure table contains the base fare structure as defined by TfL. The primary key is a composite key including the InnerZone, OuterZone, and FareTypeID columns. InnerZone and OuterZone represent the innermost and outermost zones visited. FareTypeID is a foreign key to the FareTypes table and represents a given fare type such as Adult Peak. The Fare column stores the base fare for a pair of zones and fare type.

5.2 FareTypes table

The FareTypes table contains all of the fare types defined by TfL. The FareTypeID is the primary key and is a unique integer associated with a given fare type. FareTypeName is the name or description of the fare type associated with a FareTypeID. The FareTypeID is used by the database and algorithm to identify the fare type for calculations; the FareTypeName is used primarily by the user interface.

5.3 Services table

The Services table contains the various services defined for the TfL network. ServiceID is the primary key and is a unique integer used internally by the algorithm to distinguish between services. The ServiceName is the name or description of a service and is used primarily for the user interface. Every direction of every branch of every underground line requires a separate entry in this table. Each NR route included in the model requires a separate entry. Also, every direction of every bus or tram route included in the model requires a unique entry in this table.

5.4 ServiceLinks table

The ServiceLinks table stores the station-to-station connections for the included services. The ServiceID, OriginStationID, and DestStationID columns comprise the primary key. Each entry in this table represents the existence of a physical or constructive link from one station to another via a single service. Duration stores the expected in-vehicle travel time for this link as an integer value. Similarly, distance stores the physical distance associated with this link. OriginXCoord and OriginYCoord are integer values reflecting the starting point of a link on the map currently used by the user interface. Similarly, DestXCoord and DestYCoord reflect the end point of a link on the current map. These four values may be null, and need only be defined for the user interface. The user interface will draw the shortest path on the map using the XCoord and YCoord values stored in the Stations table unless a value is defined in one of these four columns.

5.5 Stations table

The Stations table contains all of the attributes relevant to a station. The primary key is StationID, which is a unique integer assigned to a station. StationName is the name or description of the station. InnerZone and OuterZone are integer values representing the innermost zone and outermost zone a station resides in, respectively. These two values will be identical for a given station unless it is a boundary station, in which case they will be sequential. StationTypeID is an integer and foreign key to the StationTypes table. XCoord and YCoord are integers indicating the location of the station on the current map used for the user interface. DefaultTransferTime is an integer representing the time assigned to general transfers at a station. Values for transfers specified in the TransferDurations table will override the DefaultTransferTime. EntryEnabled and ExitEnabled are binary integer values indicating whether or not a station is open for passenger entry and exit, respectively. ValidatorEnabled is similarly a binary integer indicating whether or not intermediate validation may occur at a given station. HTransEnabled and HEntryExit enabled are binary values indicating whether or not the station is handicap accessible.

5.6 StationTypes table

The StationTypes table contains the various station types represented in the graphical model. StationTypeID is the primary key and a unique integer used by the algorithm to identify the station type. StationTypeName is the name or description of the station type used by the user interface. Station types are used to distinguish between bus stops, tram stops, LUL/DLR stations, and NR stations.

5.7 TransferDurations table

The TransferDurations table contains durations for transfers between two specific services at a specific station. The algorithm uses the DefaultTransferTime specified in the Stations table for all transfers at station except those transfers defined in the TransferDurations table. The primary key is a composite key including the StationID, FromServiceID, and ToServiceID columns. StationID is a foreign key in the Stations table. FromServiceID is the ID of the service one is transferring from and ToServiceID is the service to which one is transferring. Duration is an integer and represents the time associated with a given transfer. Two entries must be made in the TransferDurations table to represent two-way transfers between the same two services; these entries may have the same or different values.

5.8 Zones table

The zones table contains all of the zones currently defined within the network. The ZoneID column is the primary key and contains unique integer values representing each zone.

6 Performance

The most time-critical application of the fare calculation algorithm is implementation as part of a real-time fare collection system. In TfL's tap-in/tap-out collection system, journey fare calculations occur during the exit transaction as the passenger passes through the gate. The fare calculation cannot occur earlier because the passenger's destination is unknown until this point. In a system using a stored value smartcard, such as Oyster, the exit transaction must determine the correct fare and update the value stored on the card before the passenger removes the card from the reader. It is also reasonable to assume that a commercial payment system would require the exit transaction be complete before granting the passenger exit passage.

6.1 Performance required

The exact performance required for the fare calculation component of any fare collection system will be determined by the system's design—the portion of the total gate exit processing time allocated to fare calculation will likely be quite small. The total time available for the complete exit transaction is a function of the gate equipment and the desired passenger throughput. Typical specifications in the transit fare payment industry require total reader processing times of less than 300 milliseconds. However, work by Kocur and Maciejewski (2007) indicates TfL's requirements may be less restrictive:

> The maximum number of passengers at any gate line is approximately 35 per minute in the peak period. This allows 1.7 seconds per passenger, and reflects a typical mix of approximately 30% magnetic stripe tickets and 70% Oystercards. Magnetic stripe tickets have a performance of approximately 600 milliseconds, including the time for the passenger to insert the ticket and then remove the ticket after it is read. The passenger must perform two operations with a magnetic stripe ticket, insertion and removal, while an Oyster user only performs one operation, tapping the card.
>
> While East Asian systems may require 300 millisecond transaction times to meet their gate performance needs, it appears that TfL's needs can be satisfactorily met with cards with approximately 500 millisecond performance. East Asian systems achieve gate flows over 60 passengers per minute, in some cases approaching 100 passengers per minute. These performance levels are not required within TfL and, in high volume stations, may not be consistent with stairway and platform constraints. (Kocur & Maciejewski, 2008)

6.2 Algorithm performance

Performance tests of the adapted shortest path algorithm indicate an average calculation performance of 0.7 milliseconds. These tests were conducted using a 2GHz dual core processor and 512MB of allocated RAM. Database access time and the time for automated construction of the network were excluded from these estimates. Excluding these times is reasonable because, once initialized, the necessary network structure is stored in system memory. Each call from the fare collection system would not require reconstruction of the network; the algorithm would run on the preloaded data.

Fare calculation requiring 0.7 milliseconds is less than 0.2% of total reader processing time at 450 milliseconds, and is less than 0.3% at a processing time of 300 milliseconds. The algorithm is sufficiently fast to function even in a high volume environment.

7 Comparison of generated and current fares

The algorithm allows TfL staff to generate fares for every OD pair. This may be especially useful following a change in the zonal fare structure or a change to the network itself. Staff can compare the current fares with the new fares generated under the various path selection methods to determine the number of OD pairs that will experience a change in fare. Using historical data, TfL can estimate the number of journeys potentially affected by the fare change and the potential impact on revenue. This section uses a sample data source to generate fares for comparison with the fares published in TfL's Fare Finder.

7.1 Example data source

For the purpose of this comparison, the TfL network model was constructed from a combination of information published by TfL and information from a private website. All LUL, DLR, and London Overground stations were defined from the standard map of the underground available in most LUL stations and also from the TfL website. Service and branch information was obtained from TfL's online Journey Planner.

The base zonal fare structure was obtained from the January 2008 edition of TfL's *Your Guide to Fares and Tickets*. Current fares were retrieved from TfL's online Fare Finder application.

Duration estimates for the DLR links were obtained from Journey Planner. Duration estimates for the Underground links were obtained from a private website (http://www.geofftech.co.uk/tube/sillymaps/travel_times.jpg). A default value of six minutes was applied to all transfers. Estimates for London Overground link durations were not readily available.

Oyster journey data were averaged over a three-day period beginning Monday, 5 February 2007. The detail of these data was limited to OD pair and type of journey (pay-as-you-go or period ticket). Therefore, the following revenue analysis makes these necessary assumptions:

- demand is constant and indifferent to the fare,
- all affected journeys carry an adult peak fare, and
- daily capping never applies.

These assumptions may significantly impact the estimates of revenue changes; the revenue estimates are included for illustrative purposes only. With more detailed historical data, an analysis could better predict revenue changes resulting from changes in fare policy.

7.2 Inclusion of London Overground

The published zones and zonal fares for the London Overground are currently identical to those of the Underground and DLR. However, most of the current OD fares for the Underground and LDR were determined before the Overground became an operational part of the TfL network. If the Overground is treated as an extension of the LUL/DLR network, as implied from the standard tube map, journey fares for various OD pairs will differ from the fares currently published in Fare Finder.

Without realistic estimates of travel times on the Overground, minimum duration paths that include the Overground may be inaccurate. Thus, the comparison of OD pairs and journeys affected under the minimum duration path does not consider the effects of including the Overground.

Duration estimates do not affect fare calculation under the minimum fare path method. Duration estimates will influence path selection under this method in cases requiring tie breaking, but the fare itself will not be influenced. The minimum fare path comparison, therefore, considers the TfL network with the Overground and without.

7.3 Minimum fare path

Considering the Underground and DLR combined, there are approximately 94,000 possible OD pairs. Roughly 6% of these OD pairs do not have a current fare published in Fare Finder. The fare produced by the minimum fare path matches the published fare for 89% of the OD pairs. The algorithm produces a smaller than current fare for 3,604 OD pairs and a larger than current fare for 1,241 OD pairs.

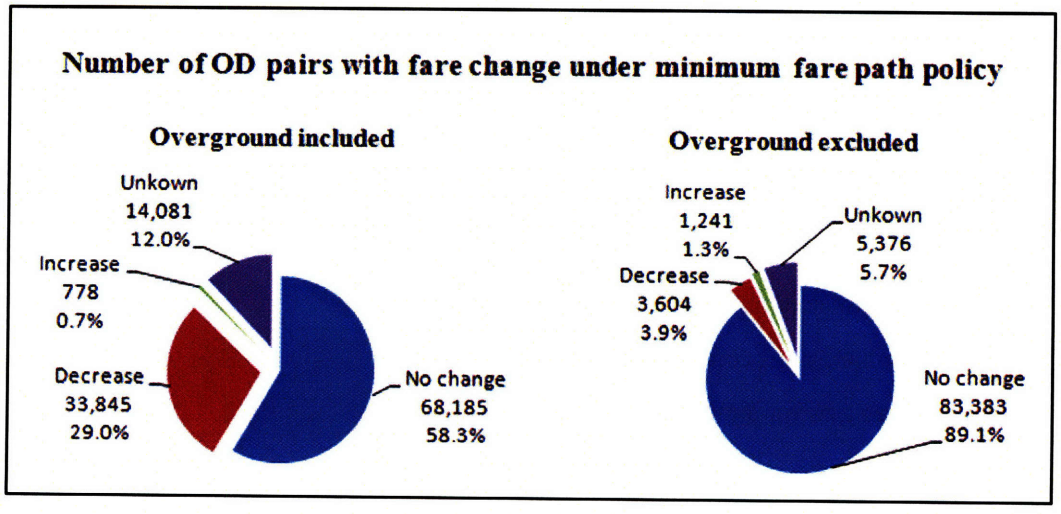


Figure 7.1: Adopting a minimum fare path fare policy affects 5-30% of all OD pairs.

Including the Overground as an extension of the Underground/DLR network adds an additional 21,000 OD pairs. Of the 117,000 total OD pairs, 12% do not have a fare published in Fare Finder. The algorithm produces a fare that matches the current fare for 58% of all OD pairs. Including the Overground provides alternate paths between many OD pairs that avoid zone 1. This results in decreased fares for a larger portion of the OD pairs; 32,513 OD pairs have a current fare higher than the minimum fare. Figure 7.1 displays the OD pair comparison for fares generate under the minimum fare path method.

Note the small number of OD pairs, roughly 1%, that have a current fare less than the calculated minimum fare. These discrepancies fall into one of two categories and typically have one station in the region illustrated in Figure 7.2. The first category includes published fares that do not reflect the current fare structure, regardless of the assumed travel path. For example, the fare published in Fare Finder for a journey from Maida Vale to Marylebone is £1.50. Maida Vale is in zone 2 and Marylebone is in zone 1, the minimum possible fare for this journey is the zone-1-to-zone-2 fare of £2.00.

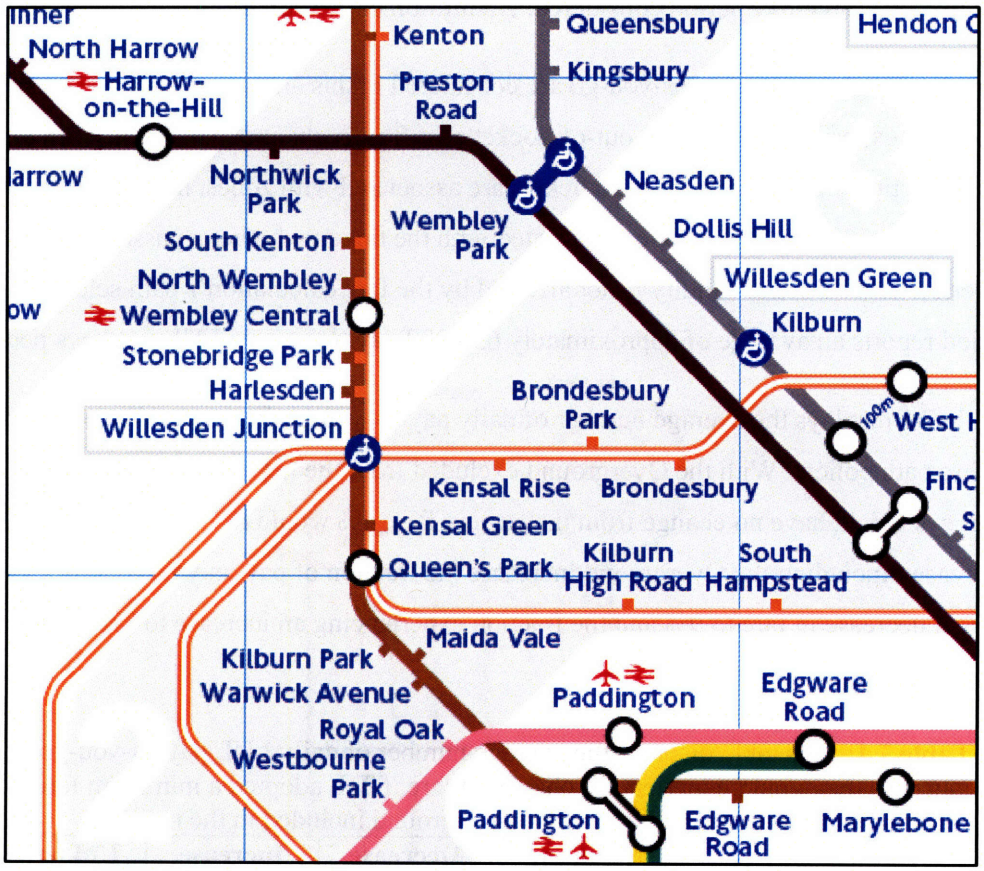


Figure 7.2: Most of the OD pairs with a published fare less than the generated minimum fare have at least one station in this region.

The second category includes published fares that either assume an infeasible path of travel or include the Overground when the algorithm has excluded it. The published fare, for example, for a journey from Harlesden in zone 3 to Harrow-on-the-Hill in zone 5 is £1.80. With the Overground excluded from the network, this journey must include travel into zone 1, and would instead be subject to a zone 1-to-zone-5 fare of £3.50.

TfL may have intentionally reduced these fares. The algorithm can replicate most of the reduced fares in the first category by including a discount for this section of the Bakerloo line. The algorithm replicates most of the reduced fares in the second category when additional service links are constructed between a zone 4 station on the Bakerloo line and a zone 4 station on the Metropolitan line.

For policy and planning purposes, the number of OD pairs affected by a fare change is arguably less important than the number of actual journeys affected. The average portion of daily Oyster journeys potentially affected by a transition to a minimum fare policy was estimated using historical Oyster data. On average over the three-day period considered, 2.5 million daily journeys were completed using Oyster.

Only passengers using pay-as-you-go are considered in this analysis. Passengers using period tickets would not experience a different out-of-pocket cost from a change in path selection rules because, under the current ticketing structure, period tickets are associated with zones, not OD pairs. A period ticket is valid for travel only in the zones associated with the ticket; whether a passenger travels outside his or her period ticket's zonal validity is not affected by the fare calculation's path selection rules. The sample period reports an average of approximately 650,000 pay-as-you-go LUL journeys per day.

Table 7.1 displays the average number of daily pay-as-you-go journeys affected by adoption of a minimum fare path policy. With the Overground excluded from the network, 94% of all LUL pay-as-you-go journeys would have no change from the current fare, 4% would have a decrease, and 1% would have an increase. Including the Overground increases the fraction of pay-as-you-go journeys experiencing a decrease in fare to 9% and the fraction experiencing an increase to 8%. The remaining 83% would have no change in fare.

Table 7.1: This table displays the average number of daily LUL pay-as-you-go journeys that would experience a change in fare if TfL adopted a minimum fare path policy, both with and without the Overground included in the network.

	No change	**Decrease**	**Increase**	**Unknown**
Overground included	565,090	64,338	51,874	3,150
Overground excluded	592,552	25,420	6,522	3,136

The historical sample can also produce estimates of the revenue changes associated with the change in fares. Given the available data and assumptions described in section 7.1, the minimum fare path policy with the Overground excluded would result in a revenue decrease of £44,500 daily. Passengers experiencing a fare increase would see, on average, an additional £0.60 per journey, on average; those experiencing a decrease would see, on average, a reduction of £1.90. Including the Overground more than doubles the revenue loss to £96,000 per day. This is a result of the larger number of journeys affected in this case. The average changes in fare are smaller than in the case when the Overground is excluded. Passengers experiencing a fare increase would see an additional £0.54 per journey; those experiencing a decrease would see an average reduction of £1.49

7.4 Minimum transfer path

The minimum transfer path method, with the Overground excluded, affects approximately the same total number of OD pairs as the minimum fare path method. The distribution of affected pairs differs, however, as the minimum transfer path results in more increased fares than decreased. As seen in Figure 7.3, 89% of OD pairs have the same minimum transfer path and current fare, 4% have a lower current fare, and 1% have a higher current fare. Even though the number of OD pairs with an unchanged fare is similar for the minimum fare path and minimum transfer path, the actual pairs may differ.

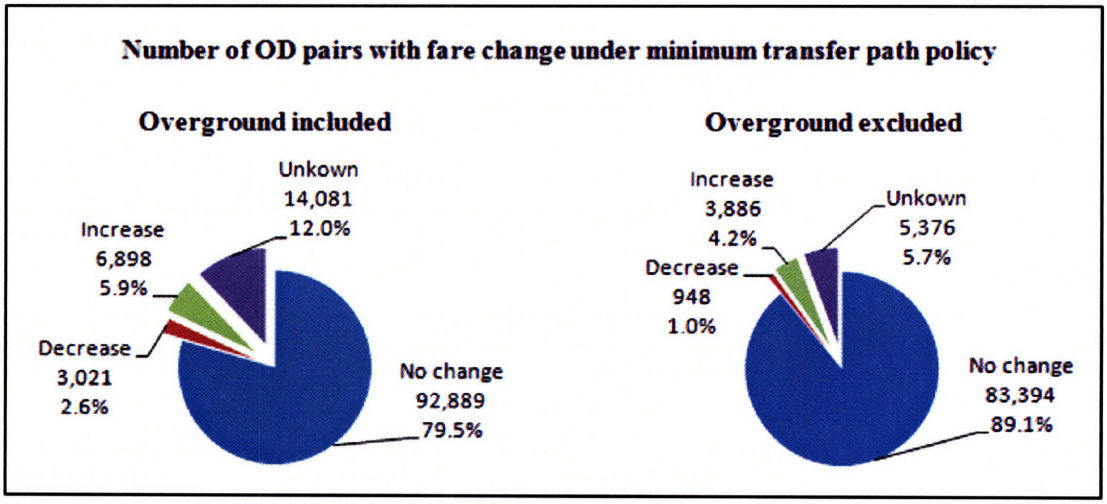


Figure 7.3: Adopting a minimum transfer path fare policy affects 5-9% of all OD pairs.

The minimum transfer path method is less likely to include the Overground in a journey than is the minimum fare path method. Thus, including the Overground in the network has less impact under the minimum transfer path than under the minimum fare path. The portion of unaffected OD pairs falls to 80%; the portion of OD pairs with a fare decrease rises to 3%; and the portion of OD pairs with a fare increase rises to 6%.

The portion of LUL pay-as-you-go journeys unaffected by the minimum transfer path method is 94% with the Overground included or excluded. This is despite the fact that an additional 10,000 OD pairs are affected when the Overground is included. Similarly, as shown in Table 7.2, the portion of journeys with an increase in fare is 2% and the portion with a decrease is 3%.

Table 7.2: This table displays the average number of daily LUL pay-as-you-go journeys that would experience a change in fare if TfL adopted a minimum transfer path fare policy, both with and without the Overground included in the network.

	No change	Decrease	Increase	Unknown
Overground included	601,101	22,245	15,398	3,150
Overground excluded	592,596	20,691	11,207	3,136

The minimum transfer path method, with the Overground included or excluded, results in a daily revenue decrease of £32,000. Passengers experiencing an increase in fare would see an additional £0.86 or £0.80 per journey with the Overground included and excluded, respectively. The average passenger experiencing a fare decrease would see a reduction of £2.05 or £1.98 with the Overground included and excluded, respectively.

7.5 Minimum duration (travel time) path

This section considers the minimum duration path only with the Overground excluded because reasonable estimates of duration for the Overground were not available. Of all the methods considered, the minimum duration path method produces the highest portion of unaffected OD pairs at 91%. Figure 7.4 shows 2% of OD pairs would increase from the current fare and 1% would decrease.

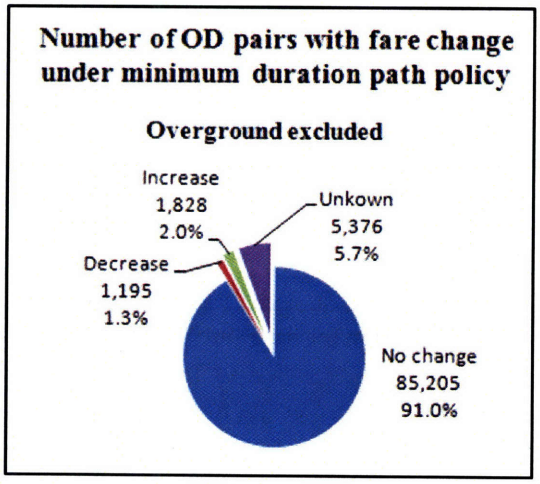


Figure 7.3: Adopting a minimum duration path fare policy affects 3% of all OD pairs.

As displayed in Table 7.3, 95% of all pay-as-you-go journeys during the sample period were between OD pairs whose fares are unaffected by the minimum duration path method. Only 1% of journeys were between pairs with a minimum duration path fare higher than the current fare, and 3% were between pairs with a lower minimum duration path fare.

Table 7.3: This table displays the average number of daily LUL pay-as-you-go journeys that would experience a change in fare if TfL adopted a minimum duration path fare policy, with the Overground excluded from the network.

	No change	**Decrease**	**Increase**	**Unknown**
Overground excluded	594,783	21,045	8,667	3,136

The minimum duration path method results in a daily revenue decrease of £39,500. Passengers experiencing an increase in fare would pay an additional £0.61 per journey. The average passenger experiencing a fare decrease would see a reduction of £2.13 per journey.

Appendix A: Current TfL fare structure

Table A.1: TfL's fare structure has nine zones and five fare types (Your guide to fares and tickets, 2008).

Zones		Zone-to-zone fare in GBP				
From	**To**	**Adult peak**	**Adult off-peak**	**16-17 New Deal peak**	**16-17 New Deal off-peak**	**5-15 all times**
1	1	1.50	1.50	0.70	0.70	0.50
n1	2	2.00	1.50	1.00	0.70	0.50
1	3	2.50	2.00	1.20	1.00	0.50
1	4	2.50	2.00	1.20	1.00	0.50
1	5	3.50	2.00	1.70	1.00	0.50
1	6	3.50	2.00	1.70	1.00	0.50
1	7	4.50	3.00	2.20	1.50	1.00
1	8	5.50	3.00	2.70	1.50	1.50
1	9	5.50	3.00	2.70	1.50	1.50
2	2	1.00	1.00	0.50	0.50	0.50
2	3	1.00	1.00	0.50	0.50	0.50
2	4	1.80	1.00	0.90	0.50	0.50
2	5	1.80	1.00	0.90	0.50	0.50
2	6	1.80	1.00	0.90	0.50	0.50
2	7	3.00	2.00	1.50	1.00	0.50
2	8	4.00	2.00	2.00	1.00	1.00
2	9	4.00	2.00	2.00	1.00	1.00
3	3	1.00	1.00	0.50	0.50	0.50
3	4	1.00	1.00	0.50	0.50	0.50
3	5	1.80	1.00	0.90	0.50	0.50
3	6	1.80	1.00	0.90	0.50	0.50
3	7	2.50	1.00	1.20	0.50	0.50
3	8	3.50	1.00	1.70	0.50	1.00
3	9	3.50	1.00	1.70	0.50	1.00
4	4	1.00	1.00	0.50	0.50	0.50
4	5	1.00	1.00	0.50	0.50	0.50
4	6	1.80	1.00	0.90	0.50	0.50
4	7	2.00	1.00	1.00	0.50	0.50
4	8	3.00	1.00	1.50	0.50	1.00
4	9	3.00	1.00	1.50	0.50	1.00
5	5	1.00	1.00	0.50	0.50	0.50
5	6	1.00	1.00	0.50	0.50	0.50
5	7	2.00	1.00	1.00	0.50	0.50
5	8	2.50	1.00	1.20	0.50	1.00
5	9	2.50	1.00	1.20	0.50	1.00
6	6	1.00	1.00	0.50	0.50	0.50
6	7	1.50	1.00	0.50	0.50	0.50
6	8	2.00	1.00	1.00	0.50	1.00
6	9	2.00	1.00	1.00	0.50	1.00
7	7	1.00	1.00	0.50	0.50	0.50
7	8	1.00	1.00	0.50	0.50	0.50
7	9	1.50	1.00	0.70	0.50	0.50
8	8	1.00	1.00	0.50	0.50	0.50
8	9	1.00	1.00	0.50	0.50	0.50
9	9	1.00	1.00	0.50	0.50	0.50

Bibliography

About ATOC. Retrieved April 10, 2008, from Association of Train Operating Companies: http://www.atoc.org/about.asp.

About us. Retrieved April 10, 2008, from Transys: http://www.transys.com/aboutus.php.

Company information. Retrieved April 6, 2008, from Transport for London: http://www.tfl.gov.uk/corporate/about-tfl/4510.aspx.

Fares and tickets supplementary information. January 2, 2008. Retrieved April 12, 2008, from Transport for London: fares-and-tickets-supplementary-info-08-01-02.

Greater London Authority Act. 1999. London, United Kingdom.

Hao, J., & Kocur, G. (1992). *A Faster Implementation of a Shortest Path Algorithm.* Waltham, MA: GTE Laboratories Incorporated.

Kocur, G., & Maciejewski, J. October 31, 2008. "Use of commercial payment media for TfL fares." Cambridge, MA.

London Buses. Retrieved April 10, 2008, from Transport for London: http://www.tfl.gov.uk/corporate/modesoftransport/1548.aspx.

London Regional Transport Act. 1984. London, United Kingdom.

London Underground. Retrieved April 10, 2008, from Transport for London: http://www.tfl.gov.uk/corporate/modesoftransport/1574.aspx.

Oyster fact sheet. January, 2008. Retrieved April 10, 2008, from Transport for London: http://www.tfl.gov.uk/assets/downloads/corporate/Oyster_Fact_Sheet_Feb_2008.pdf.

Prestige Fact Sheet. 2008. Retrieved April 10, 2008, from TranSys: http://www.transys.com/pressroom/factsheet.php.

Standard Tube Map. Retrieved May 1, 2008, from Transport for London: http://www.tfl.gov.uk/assets/downloads/Standard-Tube-map.pdf.

Your guide to fares and tickets. January 2, 2008. Retrieved April 3, 2008, from Transport for London: http://www.tfl.gov.uk/assets/downloads/guide-to-fares-and-tickets-08-01-02.pdf.