was running a mostly (but not perfectly) regular 15-min (i.e., four trains per hour) service all day from Stratford to the end of the NLL at Richmond. This service was augmented with occasional irregular services, including a shuttle that ran only as far as Camden Road, and one special that ran on the NLL to Willesden Junction but then on the West London Line to Clapham Junction. It is not immediately obvious which of these services would be attractive to a given passenger at Stratford, and thus not clear what incidence headway each passenger would experience. Earlier literature avoided this issue by avoiding stations such as Stratford altogether.

The method proposed here is designed as a tool to support the study of passenger incidence behavior in general (i.e., by including locations with heterogeneous services) and to facilitate its application in practical transit management. The method does so by estimating SWT and incidence headway automatically from the integration of published timetables with disaggregate AFC passenger journey data via schedule-based assignment.

Schedule-based assignment depends on a run-based model of public transport supply, which is very similar to the line-based model of supply, but unfolded in the temporal dimension (*11*). In such a model, each individual scheduled or actual run (or trip) of the public transport service is represented individually by its own subgraph. In the subgraph for a given run, the nodes represent the arrival, departure, or transit of that run at a specific location at a specific time. The links represent travel (or dwelling) on that run between specific points in time and space. The combination of the subgraphs of all runs is referred to as the service subgraph. Demand is also modeled with temporal as well as spatial dimensions in the demand subgraph. Nodes in this subgraph represent centroids of demand in time (according to user departure and arrival times) and space (according to the physical network). The access–egress subgraph joins the service and demand subgraphs with boarding and alighting links. The union of these three subgraphs is referred to as the diachronic graph representation. One benefit of such a representation is that shortest travel time paths can be found via standard shortest-path network algorithms such as Bellman-Ford or Dijkstra's (*12*).

In this paper it is assumed that for a given origin, destination, and time of incidence, all passengers plan to use the single schedule-based path (i.e., set of scheduled services) through the network that minimizes total travel time. Additionally, it is assumed that passengers plan itineraries to minimize the number of total boardings up to the point at which total travel time is not increased (e.g., in the trunk-and-branch example, branch-bound passengers won't board a train bound for the wrong branch just to get to the end of the trunk and transfer to the correct branch). These assumptions are necessarily a simplification of the true behaviors and perceptions of passengers. The degree to which the assumptions hold is a function of the attributes of the particular network to which they are applied and of the behavioral preferences of the passengers in question. These assumptions are sufficient to determine, for each passenger journey, the attractive departures before and subsequent to the time of incidence. SWT and incidence headway can be determined once the times of these two departures are known.

ALGORITHM AND IMPLEMENTATION

For a given passenger journey on a given public transport network, let

SWT =	scheduled waiting time for given journey,
H_I =	incidence headway for given journey,
I =	time of passenger incidence for given journey,
L_O =	location of incidence of journey in question (i.e., the origin),
L_D =	destination of that journey,
D_{prior} =	time of last attractive departure before *I,*
D_{next} =	time of first attractive departure after *I,*
$H_{\max}$ =	maximum normal headway (i.e., time between any two successive departures in same direction from same location) on network,
$H_{\min}$ =	minimum normal headway on network,
Path(from, to, time) =	function that finds shortest weighted travel time path from location *from* to location *to* with departure time strictly greater than *time,* with all travel time weights equal to 1 except for transfer or boarding penalty that is positive but less than $H_{\min}$, and
Departure(path) =	function that returns scheduled departure time of path *path.*

Equation Box 1 illustrates the algorithm to find H_I and SWT for the journey in question under the above assumptions. Lines 1 through 3 accomplish the simple task of finding the next attractive departure and thus determining SWT. Lines 4 through 9 search backward in time in increments of $H_{\min}$ until either a new attractive departure time *d* is found or the time has been moved by more than $H_{\max}$. $H_{\min}$ is the largest step possible such that the search will never skip over a possible attractive departure. In theory, the algorithm could use the timetable to determine the next departure time in this backward search process rather than blindly stepping in increments of $H_{\min}$. However, the algorithm is unaware of particular departure times since the Path( ) function encapsulates all knowledge of the timetable itself. This particular algorithmic design is motivated primarily by implementation concerns that are discussed in the following section.

A 272-line Perl script was written to implement the algorithm for individual Oyster passenger journey records. The Path( ) function

EQUATION BOX 1 Algorithm to Find SWT and Incidence Headway for a Given Passenger Journey

```
1: $p = \text{Path}(L_O, L_D, I)$
2: $D_{\text{next}} = \text{Departure}(p)$
3: $\text{SWT} = D_{\text{next}} - I$
4: $i = I$
5: $d = D_{\text{next}}$
6: while $d = D_{\text{next}}$ or $D_{\text{next}} - i \le H_{\max}$ do
7:         $i = i - H_{\min}$
8:         $d = \text{Departure}(\text{Path}(L_O, L_D, i))$
9: end while
10: if $d \ne D_{\text{next}}$ then
11:         $D_{\text{prior}} = d$
12:         $H_I = D_{\text{next}} - D_{\text{prior}}$
13: else
14:         $D_{\text{prior}} = \text{null}$
15:         $H_I = \text{null}$
16: end if
```