Transport modelling for dynamic urban climate studies: MATSDA-roads v2.0

Tiancheng Ma[1], Denise Hertwig[1], Megan McGrory[1], Matthew Paskin[1] and Sue Grimmond[1]

[1]*Department of Meteorology, University of Reading, UK*

Contents

In the following, both the main paper and this supplementary material (S prefix) are cross referenced [i.e. sections (§), Tables or Figures] with those in the main text not having a prefix of relevant numbers.

S0 Zenodo repositories

Four repositories exist for this paper (Table S1). MATSDA-roads v2.0 model codes and user manual can be found at https://doi.org/10.5281/zenodo.17736682. Processing code together with model data (MATSDA road travel database) used in this study and code for processing the Google Maps (GM) reference routes are available at https://doi.org/10.5281/zenodo.17521112. Model outputs and corresponding plotting scripts for each figure used in the main paper are at https://doi.org/10.5281/zenodo.17736562. Raw MATSDA-roads v2.0 input data are at https://doi.org/10.5281/zenodo.17736728. Details of all file types are given in Table S2 with relevant references.

Table S1: *Zenodo archive contents (https://doi.org/10.5281/zenodo.########).*

Content	Description	Reference	Zenodo archive #
MATSDA-roads v2.0	Model code and user manual	Ma et al. (2025a)	17736682
Processing codes and MATSDA travel database (London, UK)	Code includes: (i) GM routes extraction, evaluation, data cleaning, and analysis; (ii) primary input data processing to generate input to the Node Creator; (iii) MATSDA travel	Ma et al. (2025b)	17521112

	databases; (iv) MATSDA-roads Pathfinder v2.0 with adjusted input data handling for the London evaluation case. Model input data used for London case.		
MATSDA travel routes and evaluation statistics (London, UK)	Includes: (i) MATSDA-roads Pathfinder v2.0 simulation results; (ii) statistics of the MATSDA-GM routes comparison; (iii) plotting / analysis codes for figures.	Ma et al. (2025c)	17736562
Raw MATSDA-roads v2.0 input data for London, UK	Restricted access archived to allow reproducibility of model input.	Ma et al. (2025d)	17736728

***Table S2:** File types. All websites last accessed on 01/12/2025.*

Type	Description	Use	Reference
py	Python 3.11.5	Code	Python version 3.11. Available at http://www.python.org
JSON	json 2.0.9	Data format	https://www.loc.gov/preservation/digital/formats/fdd/fdd000381.shtml
shp	ESRI Shapefile	Geospatial data	https://www.loc.gov/preservation/digital/formats/fdd/fdd000280.shtml
sh	Bash script	Shell script	https://www.gnu.org/software/bash/manual/html_node/Shell-Scripts.html
GeoJSON	GeoJSON	Geospatial data	https://www.loc.gov/preservation/digital/formats/fdd/fdd000382.shtml
gpkg	GeoPackage	Geospatial data	https://www.loc.gov/preservation/digital/formats/fdd/fdd000419.shtml
csv	Comma-separated Values	Table data	https://www.loc.gov/preservation/digital/formats/fdd/fdd000323.shtml

S1 Development of MATSDA road travel database for London, UK

S1.1 Data and code overview

This section describes the specific data sources and processing steps used to generate the MATSDA-road v2.0 travel database for London. The general architecture of the MATSDA-roads v2.0 data formatting requirements are detailed in the User Manual (Ma et al. 2025a). The processing codes (Table S3) use Python 3.11.5 with `geopandas` (vn1.0.1) and are archived in Ma et al. (2025b) (Table S1). The logic of input data preparation is described in User Manual §3.1 (Ma et al. 2025a) and detailed in §S1.2. Note *MATSDA-roads_node_creator.py* is also archived at Ma et al. (2025a).

***Table S3:** Processing codes and data files of the MATSDA-roads v2.0 travel database for London*

Code / File	Type	General description	Details
1.clip_spatial_inputs.py	py	Extending junction and road data to the model domain (Fig. 1a)	§S1.2.2
2.enrich_road_segments.py	py	Map road number information from OS Open Roads (2021) to the OS Highways (2024) for road segments	§S1.2.2
3.process_speed_data.py	py	Pre-processing input speed data	§S1.2.3
4.London_private_graph_input.py	py	Processing of geospatial road network and speed input data in 500-m grid-boxes covering Greater London, UK	§S1.2.4
MATSDA-roads_node_creator.py	py	Generation of Dijkstra's nodes and graphs	§S1.3
London_500m_grid_extended.shp	shp	London processing domain out to M25 motorway	§S1.2.2
Run_Configuration_#1-7.json	JSON	MATSDA-roads v2.0 travel database (format: Python dictionary containing 'name', 'grid', 'road' and 'journeys' for nodes, while 'journeys' contain 'destination', 'time' and 'time interval') for London, UK	Table S1.9 Fig. S1.4

S1.2 Data generation

The generation of the MATSDA-roads v2.0 travel database for London with Node Creator v2.0 follows the standard workflow described in User Manual §3.1 and §3.2. This involves four steps outlined in the following sub-sections, with each having their own Python code (Table S3).

1	Pre-processing of input road network and junction data [in preparation for defining road nodes and their attributes] Code: 1.clip_spatial_inputs.py; 2.enrich_road_segments.py	§S1.2.2
2	Processing of input traffic flow speed data [*in preparation for deriving travel times on road nodes*] Code: 3.process_speed_data.py	§S1.2.3
3	Generation and harmonisation of processed input data for MATSDA's travel database [*combining outputs of steps 1 and 2*] Code: 4.London_private_graph_input.py	§S1.2.4
4	Generation of MATSDA's travel database [*using output of step 3*] Code: MATSDA-roads_node_creator.py (https://doi.org/10.5281/zenodo.17736682)	§S1.3

S1.2.1 Input Files

For Greater London, geospatial vector datasets of `LineString` objects and junction `Point` objects from the Ordnance Survey (OS Open Roads 2021, OS Highways 2024; Table S4) are used together with observed average vehicle speeds (Digimap Pilot Collection 2024).

Neighbourhood areas are grid-cells based on the MAPSECC: London (Hertwig et al. 2024, 2025a) regular Cartesian grid with a horizontal resolution of ~500 m, extended to include the important M25 circular motorway surrounding London (§*S*1.2.2 and §3.1.1, Fig. 1a).

All geospatial data use the OSGB36 (British National Grid; EPSG: 27700) coordinate reference system.

Table S4: *Data sources to generate input data for the MATSDA-road Node Creator v2.0 for London, UK.*

	Source	Type	Resolution	Version
Road segments and junctions	OS Open Roads (2021)	Vector	lines, points	2021-04
Road segments	OS Highways (2024)	Vector	lines	2023-05
500 m processing grid	Hertwig et al. (2024)	Vector	500 m, polygons	2024-03
Average traffic flow speeds	Digimap Pilot Collection (2024)	CSV	OS Highways (2024) road segments	2023-08

S1.2.2 Spatial domain and pre-processing of road input data

The *model domain* is an extension of the Hertwig et al. (2024, 2025a) MAPSECC: London 500-m grid to include the M25 motorway (Table *S*4). The resulting processing grid (Fig. 1a) includes all major traffic arteries relevant for transport across Greater London. The 500-m processing grid-cells are used for spatial consistency to typical land surface or urban weather/climate models.

In MATSDA-roads v2.0 sub-grid-scale transport processes are represented using a nodal network of connected roads.

Geospatial road network and junction location data are essential to defining road-node objects (§2.2) and their attributes (connections to other nodes via junctions, travel times as a function of traffic speed and node length, etc.) within each grid-cell. The collection of road nodes across the domain form the MATSDA-roads v2.0 travel database.

For Greater London, two Ordnance Survey (OS) road segment datasets are used (Table S4, Table S2.3) to obtain the required input data (speeds and junctions). As both datasets have the same provider, harmonising information is straightforward (e.g., through the use of common road identifiers) when needed. An example of the geospatial differences in the road network between the two input datasets is shown in Fig. S1.

Geo-referenced road `LineString` objects and junction `Point` objects (road node and motorway junction data) of two tiles (TQ and TL) of the OS Open Roads (2021) input data are extracted for the model domain (Table S3, *1.clip_spatial_inputs.py*).

Road link data are stored in the auxiliary dataset *road_link_GLA* for further processing. The Greater London OS Highways (2024) dataset is enriched with information from OS Open Roads (2021) to create the *road_number_GLA* auxiliary dataset.

Processing distinguishes four general UK road types (DfT 2012):

- local roads: low-speed minor roads (label: *'L'*)
- A roads: high-speed major roads connecting cities and regions (*'A'*)
- B roads: secondary roads linking neighbourhoods (*'B'*)
- motorways: restricted access high-speed, high-capacity roads (*'M'*)

Table S5: *Geospatial road network data sources used to generate the Greater London MATSDA-roads travel database.*

	Contains	How used?	Derived data
OS Open Roads (2021)	Junction data, carriageway type, road numbers of major roads	Determine which road nodes are connected via junctions / intersections and used for road length calculation	road_link_GLA
OS Highways (2024)	Road segments (IDs) used for speed input data	Road node definition (§2.2); calculation of weighted traffic flow speed for each road node (consisting of multiple road segments)	road_number_GLA

Although the OS Open Roads (2021) `function` attribute differentiates types of minor roads (e.g., Local Road, Secondary Access Road, Restricted Local Access Road), this is not done in the MATSDA-roads v2.0 travel database. Instead, we treat local roads within each neighbourhood as an aggregated single entity so computations remain feasible (§3.1.1).

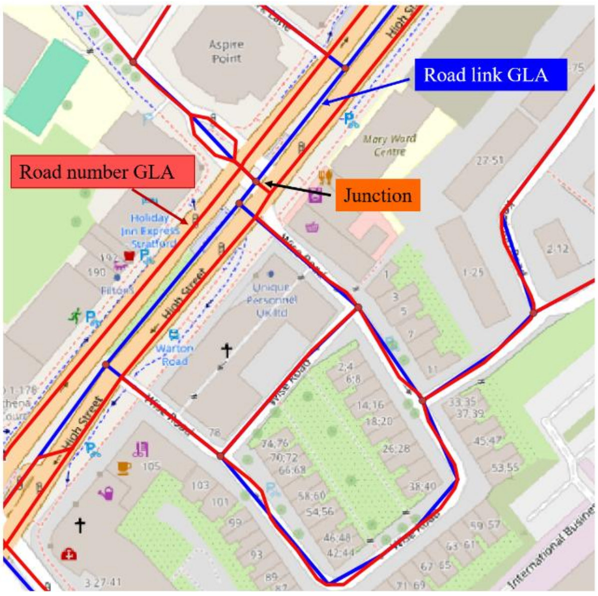

Figure S1: *Part of the London domain with OpenStreetMap raster map background, showing the two Ordnance Survey (OS) road datasets (Table S5) as road links (OS Open Roads (2021): road_link_GLA; and OS Highways (2024): road_number_GLA; colour) and junctions (OS Open Roads (2021): road_link_GLA; dots).*

To aggregate road types, we use the `featureTypeCode` attribute from the speed input data (Digimap Pilot Collection 2024; Table S6). This attribute corresponds directly to the `routehierarchy` attribute in the OS Highways (2024) dataset. Initial grouping only stratifies data based on one of the four road types with all non-major roads combined into local roads (Table S6).

Table S6: *Attributes used to group road types (*`featureTypeCode` *in Digimap Pilot Collection 2024;* `routehierarchy` *in OS Highways 2024).*

Road type	**`featureTypeCode`**	**Representation**
Motorway	3000	Motorway
A Road	3001	A Road
	4001	A Road Primary
B Road	3002	B Road
	4002	B Road Primary
Local Road	3004	Minor Road
	3007	Local Street
	4006	Local Access Road
	4009	Sec. Access Road
	4007/4010	Restricted access roads

- ❖ More road-node detail (*cf.* being solely road numbers; §3.1.1) can be includes by using the Open Roads (2021) data attribute `formOfWay` which distinguishes six carriageway types for all major road types:
  - collapsed dual carriageway (label: *'CDC'*)
  - dual carriageway (*'DC'*)
  - roundabout (*'R'*)
  - shared use carriageway (*'SUC'*)
  - single carriageway (*'SC'*)
  - slip road (*'SR'*)
- ❖ As the OS Highways (2024) road segment information has neither carriageway type nor road number, these are derived from the OS Open Roads (2021) `formOfWay` and `roadNumber` attributes, respectively.
  - To correctly assign each segment's road number, we link the OS Open Roads (2021) road segment ID attribute, `numberTOID`, to the corresponding data attribute `formspartof_href` in the OS Highways (2024) data.
  - OS Highways's `formspartof_href` has two IDs for A/B road segments (first: road name, second: road number) and only one ID for motorway segments (road number).
    - ▪ for motorways `numberTOID` from OS Open Roads (2021) matches the first ID in `formspartof_href`
    - ▪ For A- and B-roads the second ID is matched.

    - Any segments without a matching ID (i.e., local roads) remain unnumbered.
    - Data enriched OS Highways (2024) road segments are stored in a new auxiliary dataset *road_number_GLA* for further processing (Table S3, *2.enrich_road_segments.py*).
- Road junction data are derived from OS Open Roads (2021) components: *MotorwayJunction* and *RoadNode*.
  - *RoadNode* as multiple connection point types are first filtered using the `formOfNode` attribute (Table S7) to separate the data into three distinct categories: 'junction', 'pseudo node', and 'roundabout'.
    - 'pseudo nodes' typically represent points where a change in road characteristics occur, such as a transition between different carriageway types (e.g., from *'SC'* to *'DC'*).
    - *MotorwayJunction* is treated as a fourth, separate category.
  - The four distinct junction datasets (Table S7) are then spatially joined to the 500-m model domain grid (Fig. 1a) using `geopandas.sjoin`.
    - Spatial joining (using an 'inner' intersection) serves to clip the data, selecting only those junctions that are located within the model domain.
    - Four auxiliary datasets are created for subsequent processing (*junction_M25_GLA, motorway_junction_M25_GLA, roundabout_M25_GLA, pseudo_node_M25_GLA*).

Table S7: *Junction information used from the OS Open Roads (2021) primary input data.*

Junction type	**`formOfNode`**	**Junction data used**
junction (non-motorway)	junction	*RoadNode*
pseudo node	pseudo node	*RoadNode*
roundabout	roundabout	*RoadNode*
motorway junction	n/a	*MotorwayJunction*

S1.2.3 Processing of vehicle speed input data

- Traffic flow speeds are needed to determine travel durations on each road-node (§2.2) in the travel database (a function of road-node length and speed).
  - Average traffic flow speeds per OS Highways road segment obtained from the Digimap Pilot Collection (2024) were derived from observed GPS signals over 6 months (09/2023-02/2024). Data are available for different time periods and day types (Table S8).
    - Information on geographical flow direction is not given beyond two flow directions (`A` and `B`).
    - One-way roads have a single average speed (direction `A`).
    - For bidirectional roads, the code (*3.process_speed_data.py,* Table S3) selects (option) either the faster (maximum) or slower (minimum) of each pair.
    - Spatio-temporal variations of minimum flow speeds across the London data domain are shown in Fig. S2 with the maximum equivalent given in Fig. 4.
- Speed input data are created per road segment of the OS Highways dataset.
  - New data fields (pattern: <daytype_prefix>_<time_interval><flow_speed>; e.g. MF_4–7A; Table S8) are generated with the maximum/minimum flow speeds per road segment and stored.
- Data Quality Control (QC) undertaken during processing identifies erroneous speeds, mostly very high values for very short road segments (e.g., around 10 m). As node flow speeds are calculated as a length-weighted average (§3.1.2), the impact of spurious speeds on overall results is negligible.
- For subsequent processing, speeds are converted from the original units (mph or miles h^{-1}) to m min^{-1}.
- Processed speeds are mapped onto the pre-processed geospatial OS Highways (2024) road segments (*road_number_GLA* auxiliary dataset) by matching the `roadLinkID` and `TOID` attributes for speed and road datasets, respectively, which have unique IDs for each road segment.

Table S8: *Time intervals of the input speed data (Digimap Pilot Collection 2024).*

Day type	**Time interval (h, local time)**
weekday (Monday–Friday; prefix: MF)	4–7, 7–9, 9–12, 12–14, 14–16, 16–19, 19–22, 22–4
weekend (Saturday, Sunday; prefix: SS)	4–7, 7–10, 10–14, 14–19, 19–22, 22–4

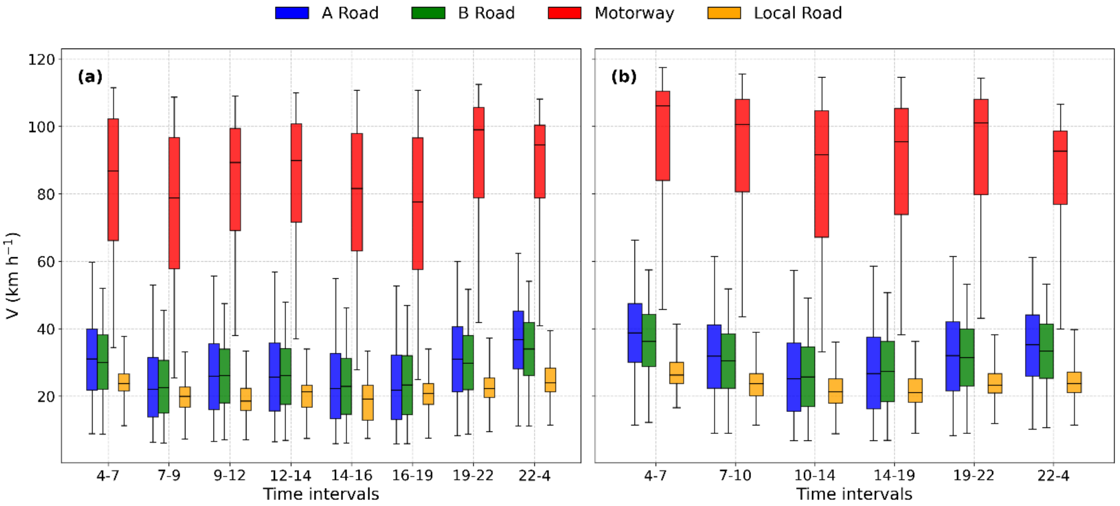

Figure S2: *London processing domain (Fig. 1c) variability of minimum traffic flow speeds (median, interquartile range, 5th and 95th percentiles) by road type (colour) and time period for: (a) weekdays and (b) weekend. Fig. 4 shows variability of maximum speeds. Data source: Digimap Pilot Collection (2024) processed as described in §1.2.3.*

S1.2.4 Merging of processed road, junction and speed data

❖ Auxiliary data generated for road, junction and flow speeds (§S1.2.2-§S1.2.3) are merged with the 500-m model domain grid.

- For each 500 m grid-cell neighbourhood, the attributes needed to generate the standardized JSON input format for the MATSDA-roads Node Creator v2.0 (User Manual Table 3.1; Ma et al. 2025a) are compiled using the code *4.London_private_graph_input.py* (Table S3).
- The output file (*MATSDA_road_database.json* under directory: *Run_Configuration_#1-7*) has the structure of a Python dictionary (User Manual Table 3.2, 3.3; Ma et al. 2025a).

S1.3 Travel database generation

Code: *MATSDA-roads_node_creator.py* (Ma et al. 2025a)

Purpose: Pre-processed road-network data (§S1.2, Table S1.8) are used to construct the final graph-structured MATSDA-roads v2.0 travel database (User Manual §3.2). Code translates grid-based road-network data (‘*MATSDA_road_database.json*’) with its information on travel time and road connectivity into a network of journeys that MATSDA’s pathfinder (§S2) can traverse using Dijkstra’s algorithm (Dijkstra 1959).

Output: The output of the processing is MATSDA’s road travel database, which for each road node holds a dictionary containing relevant attributes and a list of all possible journeys from that node (Table 3.6 in User Manual). An example of the JSON data structure is shown in Table 3.5 (User Manual §3.2.2). For each case (#1–7) considered in the main paper, a separate travel database (<[Run_Configuration] #1-7>.json) is created considering the tests and evaluation cases explored (paper Table 3).

S2 MATSDA pathfinder

The MATSDA-roads Pathfinder v2.0 (*MATSDA_pathfinder.py,* Table S9) calculates the optimal travel route through the MATSDA-roads travel database (§S1.3) for a given origin-destination grid cell pair. It loads the complete transport database, which represents all possible road segments and connections as nodes and journeys. For a given starting node (origin) and time interval, it applies Dijkstra's algorithm (Dijkstra 1959) to find the fastest route to every other node in the network (Fig. S3).

Different from the User Manual description for the general *MATSDA_pathfinder.py* (Manual §3.3; Ma et al. 2025a), for the specific London modelling case (e.g., evaluation using Google Maps routes), the input data interface of the Pathfinder is modified to consider extra information as input (e.g., geospatial information from GM). Note that modification do not affect the model core, only the way they input data are handled (e.g., full-domain/unrestricted versus restricted model run options). The modified London version (*MATSDA_pathfinder_London.py*) is available at https://doi.org/10.5281/zenodo.17521112 (Ma et al. 2025b).

Output of the MATSDA-roads Pathfinder v2.0 is a set of complete routes between an origin (start) road node and all possible destinations in the network, with one file generated for each start node. Data are organized into directories by run configuration and time interval ([Run_Configuration] #1-7/[Time_Interval]/ [routeID_StartNode].txt) for further analysis or visualisation.

Table S9: *Processing code and output of MATSDA-roads Pathfinder v2.0 for London.*

Code / File	Type	General description	Details	Zenodo
MATSDA_pathfinder.py	py	MATSDA-roads Pathfinder v2.0	User Manual §3.3	https://doi.org/10.5281/zenodo.17736682
MATSDA_pathfinder_London.py	py	MATSDA-roads Pathfinder v2.0 with input interface adjustments for the London model evaluation case.	User Manual §3.3.3 / §S2.1	https://doi.org/10.5281/zenodo.17521112
routeID_StartNode.txt	txt	MATSDA-roads Pathfinder v2.0 output (format: Python dictionary containing 'name', 'previous', 'time' and 'journey' for nodes, while 'journey' contain 'destination', 'time' and 'time interval') covering Greater London and M25 motorway, UK	User Manual Table 3.7	https://doi.org/10.5281/zenodo.17736562
df_FL_sensitivity.csv	csv	Containing 'start node', 'end node', 'GM time', 'MATSDA time', 'date', 'day type', 'hour', 'time interval', 'road type usage' and '$\mathcal{F}_L$' for each route (two directions home→work and work→home with training dataset). The training and evaluation dataset split is described in §S3.3.		
df_#1-7.csv	csv	Same as *df_FL_sensitivity.csv.* but for evaluation dataset		

S2.1 Key functions of MATSDA_pathfinder.py and MATSDA_pathfinder_London.py

a. Specific to the London case: restrict_graph()

- **Input:** graph, allowed_nodes
- **Output:** `subgraph` (dictionary)
- **Purpose**: Creates a smaller, filtered version of the main transport database. This 'sub-database' contains only a specific subset of nodes (the `allowed_nodes`) and only the journeys that travel between those nodes. Used for spatially constrained model runs (e.g., SCR cases in main paper Table 3, §3.2.3; Table S10).
- **Steps**:
    1. Filter Nodes: iterate through the full database and adds only the nodes whose names are present in the `allowed_nodes` set. Each node is created as a fresh copy with an initially empty list of journeys.
    2. Filter Journeys: iterate through the original database a second time. For each node that is in the `allowed_nodes` set, it examines its list of journeys to create a new, filtered list keeping only those journeys whose `destination` is in the `allowed_nodes` set.

3. This filtered journey list is then assigned to the corresponding node of the subsampled database.

b. General function of Pathfinder: solve_graph_for_interval()

- **Input:** start, graph, time_interval
- **Output:** `solution` (dictionary)
- **Purpose**: Finds the optimal routes based on journey weights from a single start node to every other node in the database for a specific time interval (see Fig. 3 in the main paper). This version uses the travel time as the weight and the optimisation is based on finding the shortest overall travel time using Dijkstra's algorithm (Dijkstra 1959).
- **Steps (schematic in Fig. S3)**:
  1. Initialisation of fields: `visited` (to mark nodes already processed), `travel_times` (to store the shortest time from the start node to each other node), and `solution` (to store the complete route taken). Set the travel time for the start node to 0 and for all other nodes to -1 (representing unavailability initially).
  2. Loop: Enter a loop that continues until all reachable nodes have been visited. For the `current` node, it filters its journeys to keep only those matching the specified `time_interval`. For each valid journey, it calculates the new potential travel time from the `start` node. If this new time is shorter than the previously recorded time for the destination node, it updates `travel_times` and records the path taken in the `solution` dictionary. After exploring all neighbours, the `current` node is marked as visited. Then search through all unvisited nodes to find the one with the shortest `travel_times`. This node becomes the `current` node for the next iteration of the loop.
  3. Termination: If no more unvisited or reachable nodes exist, the algorithm is complete and the loop terminates.

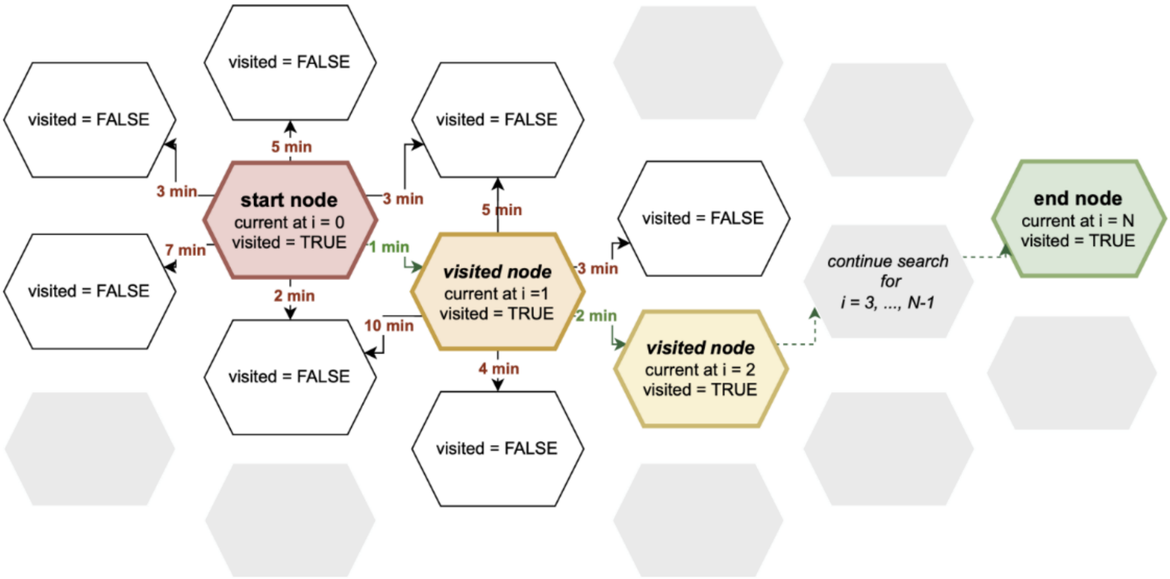

Figure S3: *Schematic of MATSDA's pathfinding sequence between a start (red) to end (green) node pair. Once all pairs have been considered, every node will have an attribute of* `visited=TRUE`*.*

S2.2 Pathfinder output

Each txt output file (one for each start node) contains a dictionary in which the keys are the destination nodes (User Manual Table 3.9). An example of the data structure is shown in User Manual Table 3.8 (§3.3.2).

S3 Processing of Google Maps (GM) reference data

To create the GM reference route dataset the selected data (§S3.1) and processed (§S3.2) codes given in Table S10 are used to obtain data from Google (2025) and are subsequently archived (Table S10).

Table S10: *Data extraction and processing codes for GM data for London. All archived at https://doi.org/10.5281/zenodo.17521112.*

Code / File	Type	General description	Details
1.Extract_GM_routes.py	Python3 code	GM API calling function	S3.1, S3.2.1
2.run_cron_Google_API.sh	Shell script	Script used to run code *Extract_GM_routes.py* through a cronjob for GM data extraction on different times of day	S3.1
3.polyline_decoder.py	Python3 code	Decodes GM API polylines and converts route data into GeoJSON format	S3.2.1
4.MATSDA_constrained_grids_by_GM_route.py	Python3 code	Identifies identical or functionally equivalent GM routes (e.g., those traversing the same grid-cell sequence) to create a non-redundant dataset of unique routes.	S3.2.2
5.assign_road_types_for_GM_route.py	Python3 code	Adding road type information for GM routes	S3.2.3
6.Divide_test_and_training_dataset.py	Python3 code	GM dataset split	S3.3

S3.1 Reference data selection

S3.1.1 Car commuter distribution

To create a dataset to evaluate MATSDA, origin-destination pairs (O-D) are chosen based on census (ONS 2011) residence-workplace spatial relations and car commuting prevalence. Figure S4 shows the number of residents per grid-cell that commute by car to a workplace within London. As car-commuter numbers are higher in the outer boroughs of London compared to the inner boroughs (Fig. S4), origin grid-cells are concentrated in the outer regions (Fig. 5). Note these numbers are only for commuters who both live and work in London.

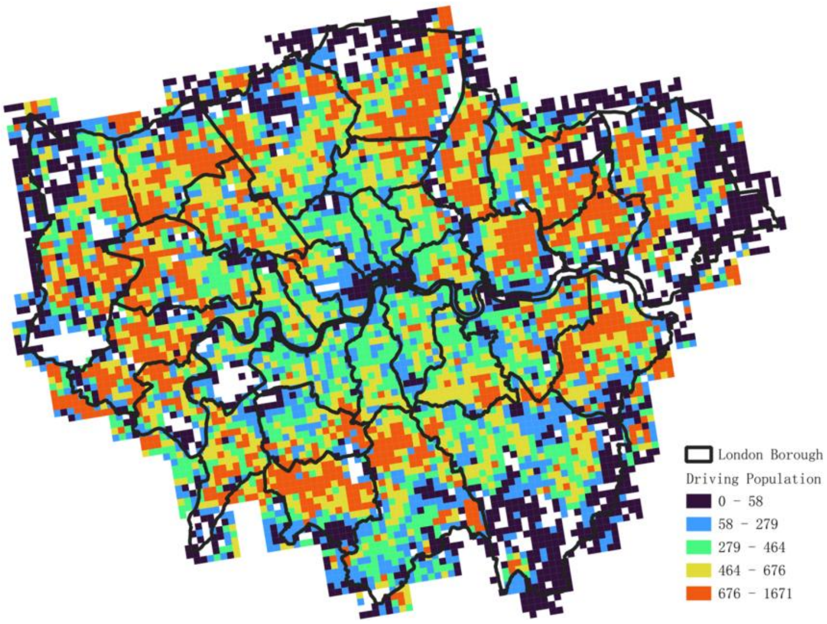

Figure S4: *Spatial distribution of car-commuter population in Greater London, who commute to a workplace within London (i.e., excludes those leaving London for work), shown per 500-m grid-cell (white: no residents). Data source: ONS (2011)*

S3.1.2 Timing of GM route extractions

Given journeys vary by time of day and day type (e.g., speed dataset, §S1.2.3), the route queries to the GM API (Google 2025) are stratified by time (Table S11). API calls are made with code *Extract_GM_routes.py*, run by shell script *run_cron_Google_API.sh* (Table S10) via a cronjob at various time (Table S11).

Table S11: *Dates and times Google Maps (Google 2025) routes were extracted via cronjob API calls at 5 min past each hour listed, and the MATSDA-road speed dataset time intervals for London.*

MATSDA (h)	Google Maps YYYY/MM/DD:	Time (h UTC)
(a)Weekday 4-7, 7-9, 9-12, 12-14, 14-16, 16-19, 19-22, 22-4	2024/12/23; 2025/02/17	3, 6, 8,11,13,15,18,21
	2024/12/24; 2025/02/18; 2025/03/26	4, 7, 9,12,14,16,19,22
	2024/12/25; 2025/02/19; 2025/03/27	5, 8,10,13,15,17,20,23
	2024/12/26; 2025/02/20; 2025/03/28	0, 6, 9,11,14,16,18,21
	2024/12/27; 2025/02/21	4, 7,10,12,15,17,19,22
	2024/12/30	1, 4, 7, 9,12,14,16,19
	2024/12/31	2, 5, 8,10,13,15,17,20
	2025/1/1	3, 6, 9,11,14,16,18,21
	2025/1/2; 2025/02/25	4, 7,10,12,15,17,19,22
	2025/1/3; 2025/02/24	5, 8,11,13,16,18,20,23
	2025/02/26	3, 6,11,17,21
	2025/02/27	2, 8,12,16,18,21
	2025/02/28	1, 7,10,13,17,22
(b) Weekend 4-7, 7-10, 10-14, 14-19, 19-22, 22-4	2024/12/28; 2025/02/22; 2025/03/30	5, 8,11,13,15,17,20,22
	2024/12/29; 2025/02/23	6, 9,12,14,16,18,21,23
	2025/03/29	4, 7,10,12,14,16,19,21

S3.2 Reference data pre-processing

S3.2.1 Raw data processing

The raw data obtained from the GM Directions API (Google 2025) is in JSON format, with an encoded polyline string per route segment.

Data are converted into geospatial format compatible with MATSDA's grid system using the code *polyline_decoder.py* calling the **google_maps_geojson()** function (within code *Extract_GM_routes.py*, Table S10) for each route, processing each route segment.

For each segment, **decode_polyline()** decodes the retrieved encoded polyline, by using a list of coordinate pairs (latitude and longitude coordinates) to create a GeoJSON `LineString` feature, adding the following attributes from the GM data:

1. travel duration (in seconds)
2. segment length (in metres)

The resulting collection of features forms a complete, geographically accurate representation of the GM route and is saved as a GeoJSON file for further processing. Output format:

GM_route_<origin>_<destination>_driving_best_guess_<alternative>_<time>.geojson

An example: GM_route_8270201_22310203_driving_best_guess_0_2024_12_25_10_17_07.geojson

S3.2.2 Derivation of unique routes

As the API requests (Table S11) often return identical or functionally equivalent (e.g., traverse same grid cells) routes, redundant data are removed using the code *MATSDA_constrained_grids_by_GM_route.py* (Table S10), using the following steps:

a) Each GM route is assigned a unique identifier (ID) which captures four key attributes in this format:
`route_<O-D pair ID>_<Alt ID>_<Day ID>_<Time ID>`

`O-D pair ID`	Unique index for each origin-destination (O-D) pair
`Alt ID`	Index for specific route geometry (primary or alternative) for O-D pair
`Day ID`	Day type: 0 weekday (WD$_{NS}$, WD$_{S}$); 1 weekend or holiday (WE/H).
`Time ID`	Time of day: Weekdays: 0-7 for the eight MF_* intervals (Table S8) Weekends: 0-5 six SS_* intervals

b) When a new route is assigned an existing ID, the travel duration is compared to the stored record

Duration difference negligible (< 0.0001 min)	New route discarded as a duplicate
Duration difference non-negligible	New duration and corresponding timestamp are appended to the existing record

This allows the temporal variability for single routes to be captured.

c) Different route alternatives are sometimes functionally identical at the MATSDA grid resolution (500 m x 500 m).

If two routes with different `LineString` geometries are found to traverse the exact same sequence of grid cells, they are considered duplicates and merged into a single unique route entry. This ensures only meaningfully distinct paths are retained for the final analysis.

The output of the steps above is a GeoPackage database file and a GeoJSON file for each unique GM route with attributes listed in Table S12.

Table S12: *Dataset attributes for each unique GM route within the GPKG (GeoJSON) file above (below) the horizontal black line.*

Attribute name	**Units**	**Type**	**Description**
unique_route_id	-	integer	ID for unique O-D pair
alt_index	-	integer	ID for different route alternatives w.r.t. a specific O-D pair
day_type_code	-	integer	ID for the day type
time_interval_code	-	integer	ID for the time of day
durations	min	string	Travel time for the unique route
date_hours	-	string	Extracted date and time of day (h)
source_file	-	string	Originated GM route data
grid_id	-	string	Grids of the unique route traversed

S3.2.3 Derivation of road type fractions

To enable a like-for-like comparison between GM routes and MATSDA's road network, each segment of a unique GM route is classified according to the OS Highways (2024) road data, using the code *assign_road_types_for_GM_route.py* (Table S10), that includes the following steps:

1. *Buffering*: Each GM route segment (`LineString` objects) is buffered by 4 m creating a narrow polygon to account for minor geospatial alignment discrepancies between GM routes and OS Highways (2024) road network datasets.
2. *Spatial joining:* The buffered GM segments and OS Highways (2024) roads are joined (`geopandas.sjoin`), with the `sjoin` predicate '`contains`' used to identify the OS road types falling within each buffered GM segment.
3. *Majority rule classification*: As a single buffered segment may overlap with multiple OS road types (especially at junctions), a 'majority rule' is applied. The code counts how many times each OS road type is associated with a single GM segment and assigns the most frequently occurring type.
4. *Gap filling:* As some segments (e.g. very short ones, parts of spatially complex intersections) remain unclassified, gap-filling is undertaken to ensure continuity.

If preceding and succeeding segments have the same road type	→ assign this road type
If not	→ assign 'Local Road'

5. *Final output*: A geospatial file for each unique GM route, with every segment having a specific road type. This allows calculation of road type usage ratios (i.e., f_k Eq. 4) and comparison with MATSDA's routing outputs (Fig. S5, 14).

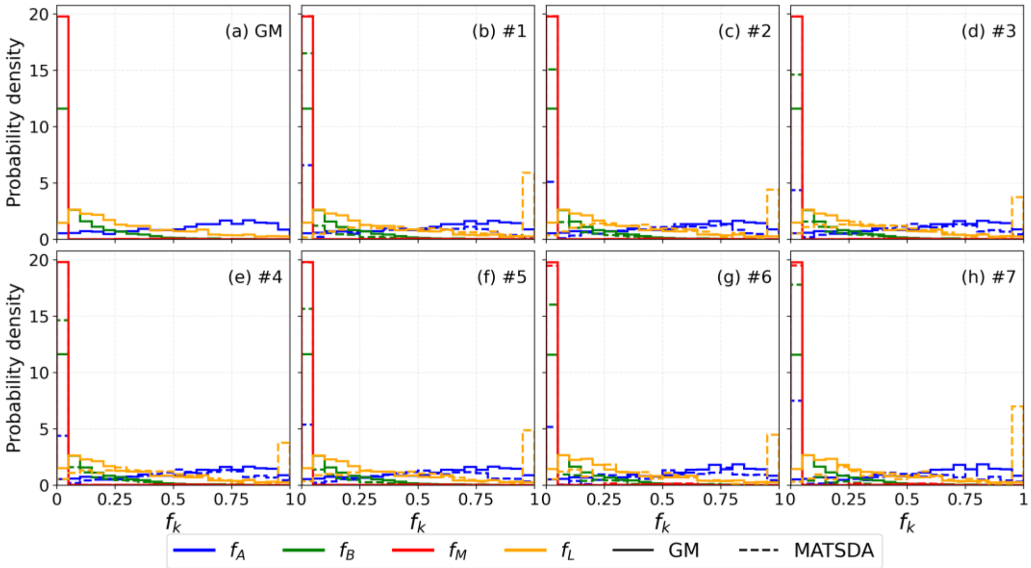

Figure S5: *Probability density of road type fractions (f_k with k=A, B, M, L, Eq. 4) of A (f_A, blue)/B (f_B, yellow)/local (f_L, red) roads and Motorways (f_M, green) derived from: (a) the GM reference data and (b-h) MATSDA #1-7 for journeys ⩽14 km. Line styles distinguish between GM (solid) and MATSDA cases (dashed).*

S3.3 Splitting of GM dataset

The GM routes are categorized based on four key characteristics (Table S13): (1) day type, (2) time of day, (3) travel distance, and (4) dominant road type.

Of the 44,841 unique routes retrieved (§S3.2.2), 8,205 routes are excluded from the travel time analysis:
(a) 739 routes are partially outside the model's geographical domain boundary,
(b) 7,466 routes cannot be assessed due to constraints in the underlying road data, preventing a direct comparison to MATSDA.
This leaves a total of 36,636 routes for the comparison of travel times (temporal analysis) (Table S13).

Table S13: *Number of journeys in the GM dataset (Table 2) including all sample Total (including those which went beyond out model domain), Full domain (restricted to our model domain), which are then randomly split into two independent data sets for Training and Evaluation (with ratio indicating the fraction of training within each subclass) by (**a**) journey distance and (**b**) dominant road type (longest part of the journey),(**c**) by day type and (**d**) time of day.*

	Number of journeys				Ratio
	Total	**Training**	**Evaluation**	Full	***Training***
a. Distance	**36,636**	**21,969**	**14,667**	**22,133**	
Long (L) > 29 km	15686	9487	6199	12688	0.605
Medium-Long (ML) 21 to ≤ 29 km	7016	4160	2856	3404	0.593
Medium (M) 13 to ≤ 21 km	6079	3617	2462	2753	0.595
Short (S) ≤ 13 km	7855	4705	3150	3288	0.599
b. Dominant road type					
A Road	32153	19284	12869	16783	0.6
B Road	470	268	202	202	0.57
Local Road	3401	2053	1348	1358	0.604
Motorway	612	364	248	3790	0.595
c. Day type					
Holiday (H)	5506	3301	2205	3449	
Weekend (WE)	7522	4510	3012	4301	
Weekday with school (WD_S)	7067	4237	2830	3777	
Weekday- no School (WD_NS)	16541	9921	6620	10606	
d. Time of day					
Weekday (WD) MF_4-7	2562	1536	1026	1467	
MF_7-9	2846	1707	1139	1777	
MF_9-12	2818	1690	1128	1690	
MF_12-14	2942	1765	1177	1807	
MF_14-16	2707	1623	1084	1695	
MF_16-19	3845	2306	1539	2397	
MF_19-22	2930	1757	1173	1836	
MF_22-4	2958	1774	1184	1714	
Weekend (WE) SS_4-7	1989	1192	797	1142	
SS_7-10	1877	1125	752	1127	
SS_10-14	2376	1425	951	1414	
SS_14-19	2844	1706	1138	1709	
SS_19-22	2022	1212	810	1213	
SS_22-4	1920	1151	769	1145	

The stratified splitting methodology uses the code *Divide_test_and_training_dataset.py* (Table S10) to ensure the training and evaluation datasets are both representative. Routes are grouped in the following order:
(a) by day type (e.g., weekday, weekend)
(b) sub-set by time of day (e.g., MF_7-9)
From the subgroups, routes are randomly allocated to 60% for Training and the remaining to the Evaluation set (ratio, Table S13) to preserves the distribution of routes across conditions in both datasets.

The final datasets to train and evaluate MATSDA from the GM route dataset are (Table S13):
1) *Training*: used for model parameter sensitivity tests
2) *Evaluation (temporal)*: used for travel time performance assessment.
3) *Evaluation (spatial)*: includes the temporal evaluation plus the 7,466 routes initially excluded due to GM constraints, allowing for a broader performance check (both temporal and spatial) that includes routes beyond just those suitable for a direct travel-time assessment.

S4 Parametrization of effective local road length fraction ($\mathcal{F}_L$)

The effective local road length fraction ($\mathcal{F}_L$) parameterization (discussed in §4.1.1) methodology uses analysis of the training dataset (§S3.3, Table 2) to ensure subsequent independent model evaluation and its variations (e.g., by number of junctions or distance). This is part of the input data generation (User Manual §3.1.1, Ma et al. 2025a)

S4.1 Data and definitions

To derive $\mathcal{F}_L$, two key length scales are required for each 500 m grid-cell neighbourhood traversed by the training Google Maps (GM) routes:

- Total available local road length (l_L): total physical length of all local road segments within a neighbourhood, as stored in the MATSDA travel database (derived in §S1.3). This static pre-calculated value represents the maximum possible distance of travel on local roads within each neighbourhood.
- Local road length used (λ_L): actual distance travelled on local roads within a neighbourhood for a specific route, derived from the GM reference data (derived in §S3.2.3).

These two metrics are used to calculate $\mathcal{F}_L$ for each traversed neighbourhood, as defined by Eq. (3): $\mathcal{F}_L = \frac{\lambda_L}{l_L}$.

Grid-cells with:

- ❖ $\mathcal{F}_L = 0$ are excluded from further analysis
- ❖ $\mathcal{F}_L > 1$ can occur due to small discrepancies in road classification between the geospatial GM routes and the geospatial representation of roads in the OS Highways (2024) dataset. These instances (1% of cases) are also excluded from further analysis.

S4.2 Methodology

S4.2.1 Dependency on number of junctions

The number of junctions between road types per grid-cell increases as $\mathcal{F}_L$ decreases (Fig. S6). This suggest using a constant $\mathcal{F}_L$ value may be inappropriate across all routes, as it would overestimate local road usage in grid-cells with complex local road networks with many junctions providing more options, with reduced total distance. However, an $\mathcal{F}_L$ parameterization based on number of junctions is not implemented as this information is unknown before an exact route is selected and therefore impractical for the MATSDA-roads v2.0 Pathfinder as the continuous updating would create a large computational overhead.

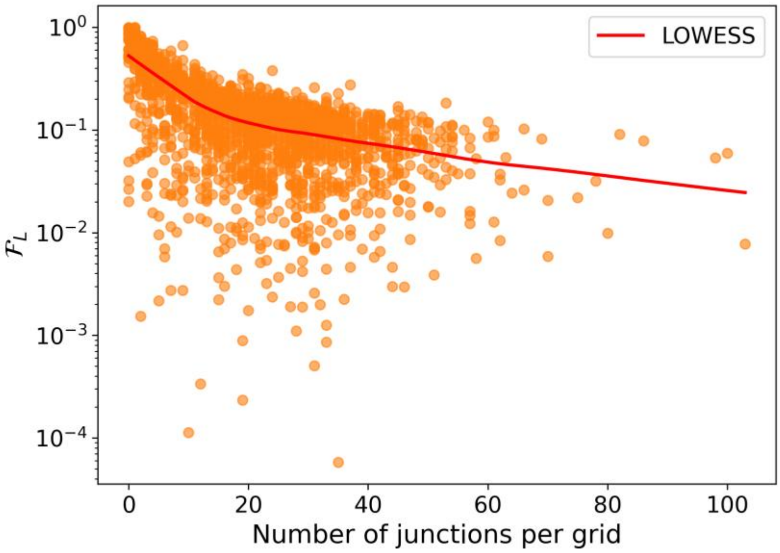

Figure S6: *$\mathcal{F}_L$ (log-scale axis) as a function of the total number of local, A- and B-road junctions per grid-cell with LOWESS curve (line, Cleveland 1979). Derived from road junction data junction_M25_GLA.shp, pseudo_node_M25_GLA.shp (§S1.2.2) with roundabouts and motorway junctions excluded to improve the local road focus.*

S4.2.2 Dependency on route length

With a junction-based $\mathcal{F}_L$ parameterization being impractical (§S4.2.1), we consider the importance of local roads relative to major roads travel as behaviour can vary with route length (Stead and Marshall 2001). To make MATSDA's $\mathcal{F}_L$ responsive to route length, we use the straight-line Origin-Destination (O-D) distance (d, unit: m; measured between grid-cell centres), a metric available before the detailed route is computed.

The distance-based parameterization has five steps:

1) *Perform sensitivity runs with constant* $\mathcal{F}_L$. Nine runs (Run#3, Table 3) each using a different, but fixed $\mathcal{F}_L$ value (x-axis, Fig. S7) for the training set (Table 2) O-D pairs, creating a predicted travel time dataset.
2) *Statistical analyses at route-level*. Statistical analysis of the thousands of individual routes in the travel times dataset (1) by unique O-D pair and time periods (Morning Rush, Morning Off-Peak, Evening Rush, Night Off-Peak) provides the median MATSDA and GM travel times.
3) *Aggregate O-D pairs by straight-line distance.* The O-D pair statistics (2) are aggregated into 5 km straight-line distance bins (starting from (1.5-5] km, then (5-10] km etc., Fig. S7) across the 1.5 to 45 km dataset range to ensure all have more than 500 samples (bins < 35 km have >1000).
4) *Identify optimal* $\mathcal{F}_L$ *for each distance bin*. Across the nine distance bins and four time periods, the mean absolute error (MAE; Eq. 6) of the median MATSDA travel times (*cf.* GM) per $\mathcal{F}_L$ run is obtained. The smallest MAE indicates the optimal $\mathcal{F}_L$ by distance bin and time period (Fig. S7).
5) *Synthesize results*. Analysis of the optimal $\mathcal{F}_L$ values across all distances shows:

- For shorter journeys (Fig. S7a-c) the optimal $\mathcal{F}_L$ values are consistently higher (*cf.* longer distances) across all time periods.
- For longer journeys (Fig. S7d-i) optimal $\mathcal{F}_L$ values are smaller, dropping to 0.14 or 0.1 (i.e., distribution shift of minimum MAE).

Analysis of GM journeys (Fig. 7d-f) suggests the local road length fraction used is slightly larger for shorter journeys which take less time. Given the inner-city speed limit of 20 miles h^{-1} (TfL 2023, Fig. S10) or ~32 km h^{-1} on local roads, the maximum distance that could be travelled in 30 min is <16 km. The analysis of model runs (Fig. S7) indicate a change in optimal $\mathcal{F}_L$ at ~14 km O-D straight-line distance. Hence, for Run #4, #5, and #7 (Table 3) we assess the use of $\mathcal{F}_L$=0.16 (for O-D straight-line distances ≤ 14 km) and $\mathcal{F}_L$=0.14 (> 14 km), assuming a swifter connection is made to faster, major arterial roads for longer trips. A comparison of traffic speeds derived from the GM reference data and MATSDA-roads input data is shown in Fig. S11. While the MATSDA traffic speed data captures distinct rush-hour speed reductions, the GM-derived speeds exhibit smaller diurnal variations across the day.

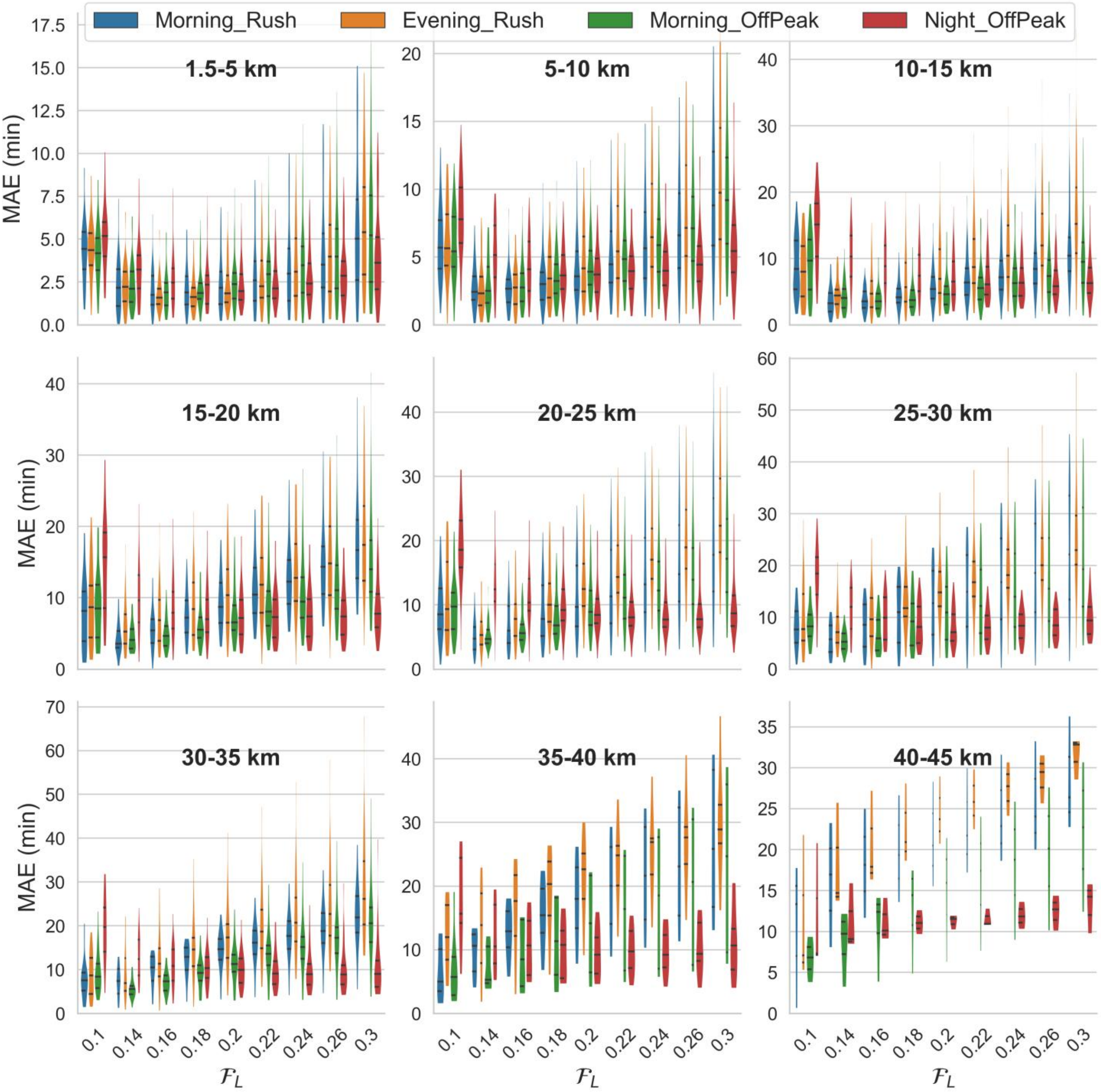

Figure S7: *The optimal* $\mathcal{F}_L$ *(Run #3, Table 3) (i.e. lowest median mean absolute error, MAE) for (***a-i***) nine O-D straight-line distance (panels) by time periods (colour), relative to the MAE variation (Y-scales vary between sub-plots) for a range of* $\mathcal{F}_L$ *values (x-axis, note scale not linear).*

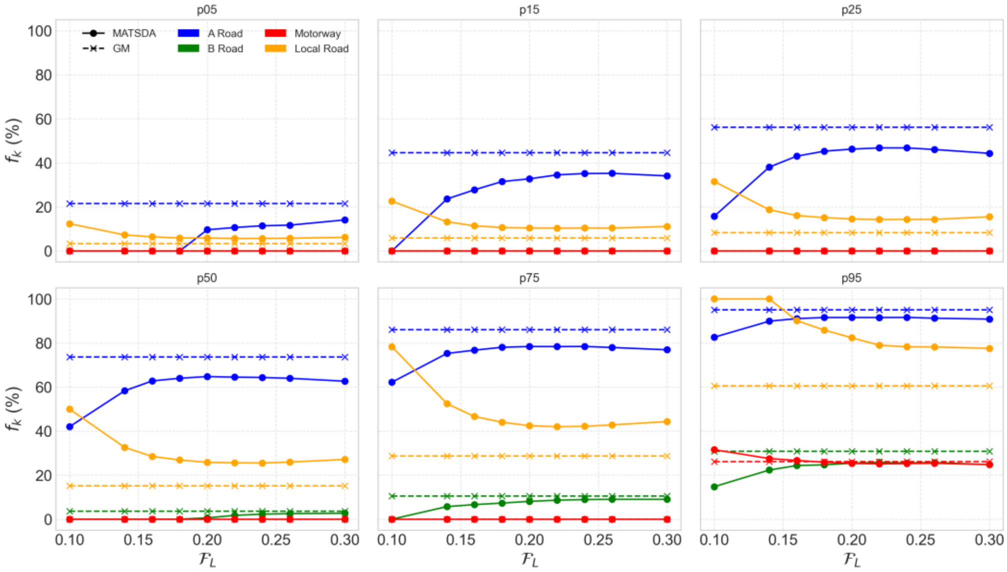

Figure S8: *Road-type usage ratio ($\boldsymbol{f_{k,MATSDA}}$ and $\boldsymbol{f_{k,GM}}$, Eq. 4) as a function of prescribed effective local road length fraction ($\boldsymbol{\mathcal{F}_L}$) for different road types (colour; A/B/L/M) from two sources (lines type; GM/MATSDA) for WDs day type (run#3 Table 3) with both data sorted to show behaviour at six percentiles of the distribution (subplots p: 5, 15, 15, 50, 75, 95).*

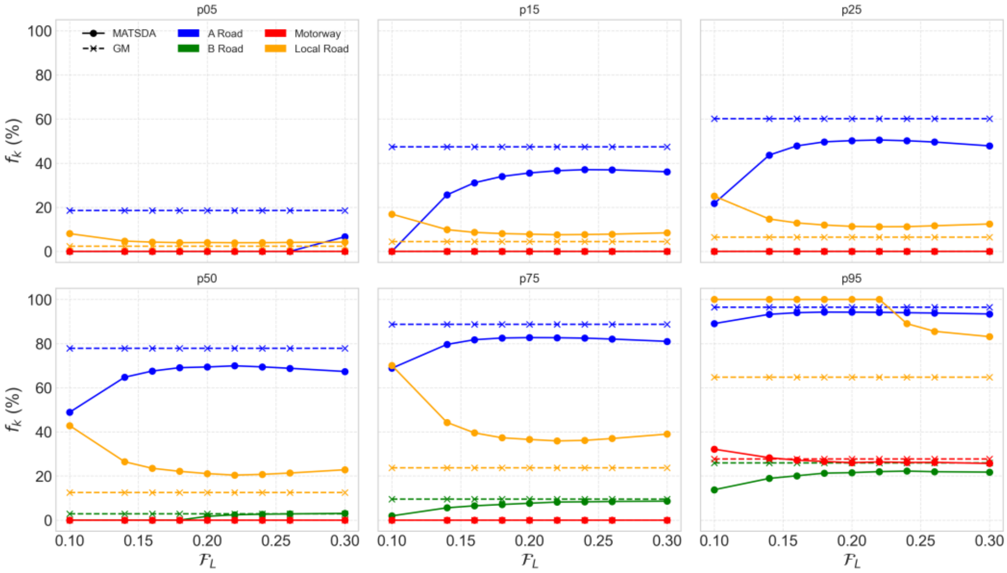

Figure S9: *Road-type usage ratio ($\boldsymbol{f_{k,MATSDA}}$ and $\boldsymbol{f_{k,GM}}$, Eq. 4) as a function of prescribed effective local road length fraction ($\boldsymbol{\mathcal{F}_L}$) for different road types (colour; A/B/L/M) from two sources (lines type; GM/MATSDA) for H / WE day type (run#3 Table 3) with both data sorted to show behaviour at six percentiles of the distribution (subplots p: 5, 15, 15, 50, 75, 95).*

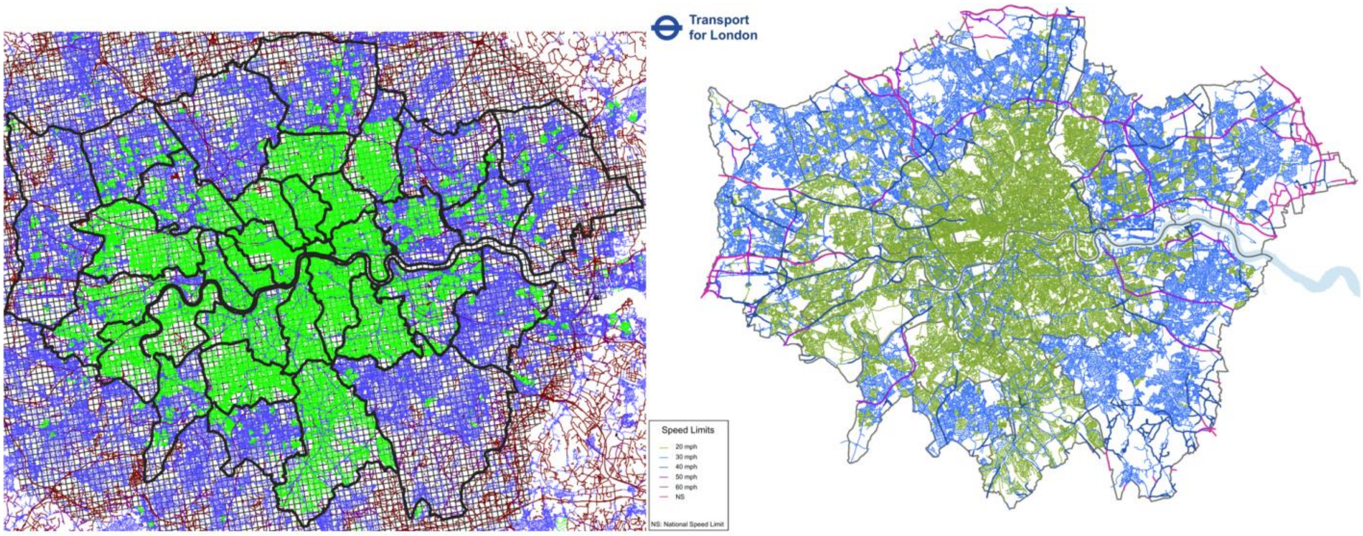

Figure S10: *Road speed limits in Greater London (note: colour bar of both in miles per hour (mph) to be consistent with b) as (**a**) set in the MATSDA input data (§S1.2.4), and (**b**) the actual values given by Transport for London (2023).*

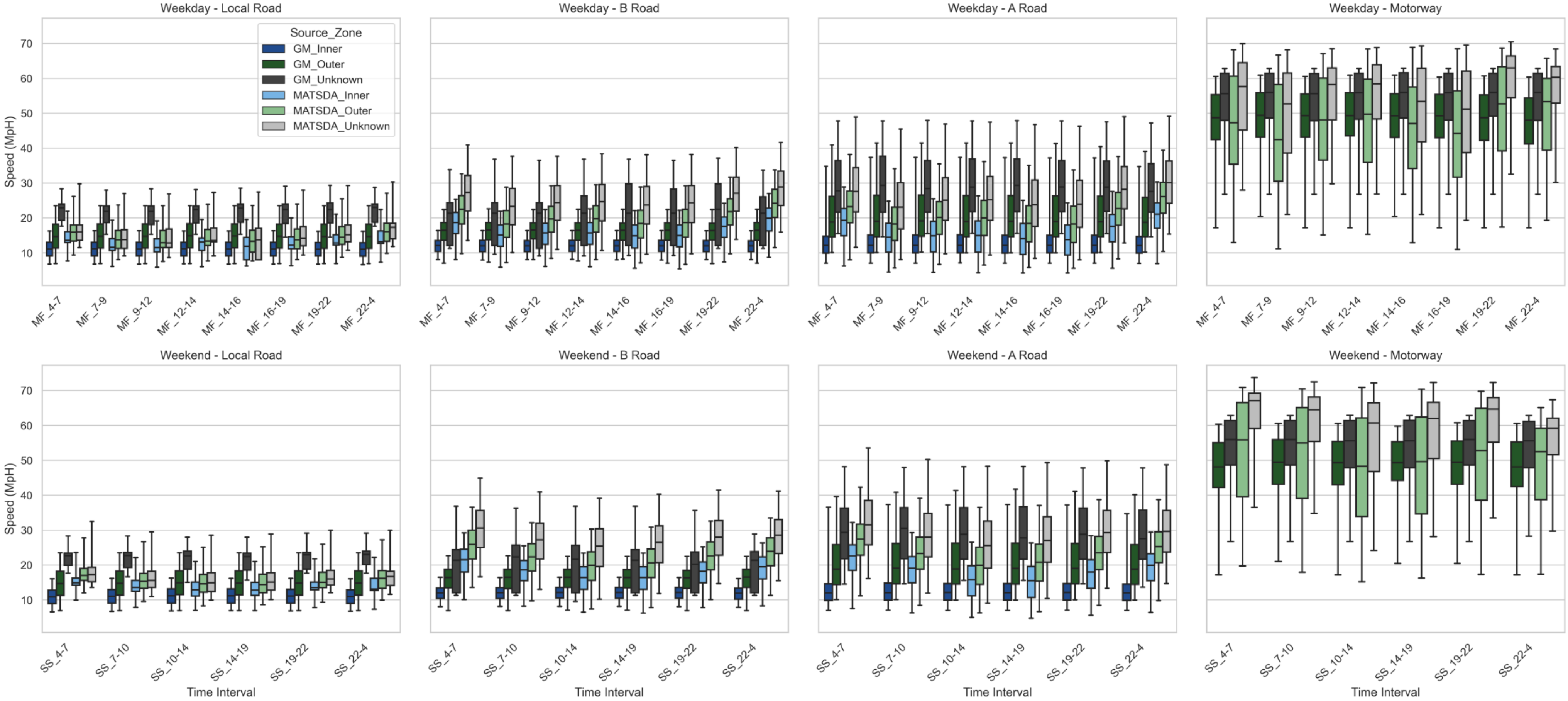

Figure S11: *Variation of traffic speeds by time of day for inner and outer Boroughs (Fig. S10) on (**a-d**) weekday and (**e-h**) weekend for **(a.d)** local **(b,c)** B (**c,f**) A roads and (**d,h**) motorways, from MATSDA and the GM (Table 2-Training).*

References

Capel-Timms, I., Smith, S. T., Sun, T., and Grimmond, S.: Dynamic Anthropogenic activitieS impacting Heat emissions (DASH v1.0): development and evaluation., Geosci. Model Dev., 13, 4891–4924, https://doi.org/10.5194/gmd-13-4891-2020, 2020.

Cleveland, W. S.: Robust Locally Weighted Regression and Smoothing Scatterplots., J. Am. Stat. Assoc., 74, 829–836, https://doi.org/10.1080/01621459.1979.10481038, 1979.

DfT: Guidance on road classification and the primary route network., [online] Available from: https://www.gov.uk/government/publications/guidance-on-road-classification-and-the-primary-route-network/guidance-on-road-classification-and-the-primary-route-network (Accessed 2 July 2025), 2012.

Digimap Pilot Collection: Average Speeds, Scale 1:10000, Tiles: GB, Updated: 31 August 2023, Basemap, Using: EDINA Pilot Digimap Service, [online] Available from: https://digimap.edina.ac.uk (Accessed 1 March 2024), 2024.

Dijkstra, E. W.: A note on two problems in connexion with graphs., Numer. Math., 1, 269–271, https://doi.org/10.1007/BF01386390, 1959.

Google: Google Maps Directions API., [online] Available from: https://developers.google.com/maps/documentation/directions (Accessed 23 December 2024), 2025.

Hertwig, D., McGrory, M., Paskin, M., Liu, Y., Lo Piano, S., Llanwarne, H., Smith, S. T., and Grimmond, S.: Multi-scale harmonisation Across Physical and Socio-Economic Characteristics of a City region (MAPSECC): London, UK., Zenodo, https://doi.org/10.5281/zenodo.12190341, 2024.

Hertwig, D., McGrory, M., Paskin, M., Liu, Y., Lo Piano, S., Llanwarne, H., Smith, S. T., and Grimmond, S.: Connecting physical and socio-economic spaces for multi-scale urban modelling: a dataset for London., Geosci. Data J., https://doi.org/10.1002/gdj3.289, 2025a.

Hertwig, D., McGrory, M., Liu, Y., Ma, T., Paskin, M., Smith, S. T., and Grimmond, S.: DAVE (Dynamic Anthropogenic actiVities and feedback to Emissions) Documentation and Manual (2025_1.0)., Zenodo, https://doi.org/10.5281/zenodo.11369893, 2025b.

Ma, T., Hertwig, D., McGrory, M., Paskin, M., & Grimmond, S., MATSDA-roads (Movement And Transport Simulations using Dijkstra's Algorithm - roads) (v2.0). Zenodo. https://doi.org/10.5281/zenodo.17736682, 2025a

Ma, T., Hertwig, D., McGrory, M., Paskin, M., & Grimmond, S., Model input and processing codes for "Transport modelling for dynamic urban climate studies: MATSDA-roads v2.0" (v1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.17521112, 2025b

Ma, T., Hertwig, D., McGrory, M., Paskin, M., & Grimmond, S., Model output and plotting codes for "Transport modelling for dynamic urban climate studies: MATSDA-roads v2.0" (v1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.17736562, 2025c.

Ma, T., Hertwig, D., McGrory, M., Paskin, M., & Grimmond, S., Raw input data for "Transport modelling for dynamic urban climate studies: MATSDA-roads v2.0" [Data set]. Zenodo. https://doi.org/10.5281/zenodo.17736728, 2025d.

ONS: Office for National Statistics 2011: Location of usual residence and place of work (with outside UK collapsed) (OA/WPZ level)., [online] Available from: https://www.nomisweb.co.uk/census/2011/wf02ew (Accessed 10 March 2023), 2011.

Ordnance Survey: OS MasterMap Highways Network [GML3 geospatial data], Scale 1:2500, Tiles: GB, Updated: 26 May 2023, Using: EDINA Digimap Ordnance Survey Service, [online] Available from: https://digimap.edina.ac.uk (Accessed 9 April 2024).

OS Open Roads: Ordnance Survey Open Roads dataset. Version date: 2021-04., [online] Available from: https://osdatahub.os.uk/downloads/open/OpenRoads (Accessed 16 March 2024), 2021.

Python Software Foundation. Python Language Reference, version 3.11. Available at http://www.python.org

Stead, D. and Marshall, S.: The Relationships between Urban Form and Travel Patterns. An International Review and Evaluation., Eur. J. Transp. Infrastruct. Res., 1, https://doi.org/10.18757/ejtir.2001.1.2.3497, 2001.

Transport for London: London Digital Speed Limit Map., [online] Available from: https://content.tfl.gov.uk/london-digital-speed-limit-map.pdf, 2023.